Invitation to Formal Semantics

(Formerly known as *Semantics Boot Camp*)

Elizabeth Coppock
&
Lucas Champollion

Draft of August 26, 2026

Preface

Semantics, the study of meaning, is a core subfield of linguistics, a discipline that integrates methods from the social sciences, liberal arts, and mathematics to study the nature of language. Since the 1970s, much of semantics has taken a formal turn, including techniques from mathematics such as logic and set theory. This textbook is a gentle and compact introduction to these techniques, and focuses on the way the meaning of individual expressions in natural language (words and phrases) combine to produce larger meaningful expressions such as sentences and texts.

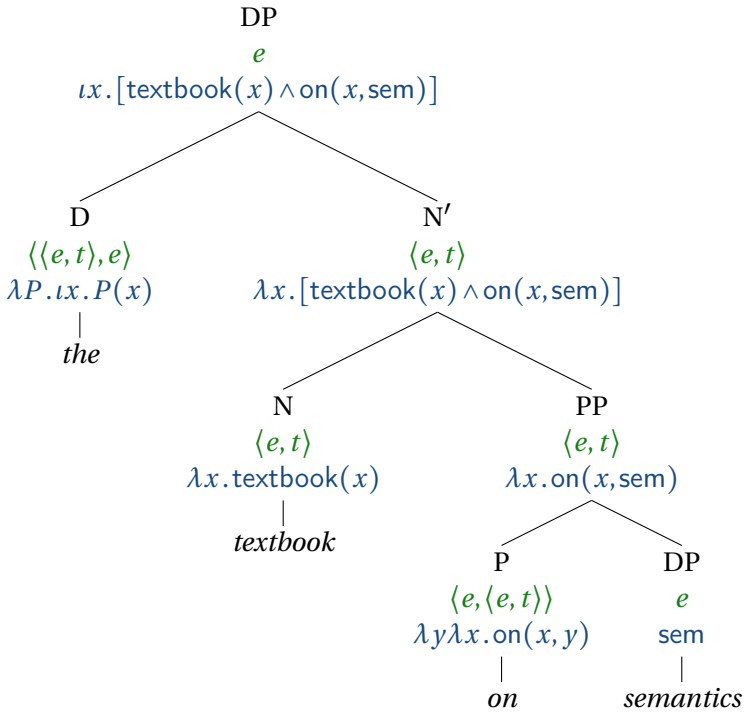
Students are guided through the development of a formally precise, compositional, model-theoretic account of semantics, using a logical representation language that is well-rooted in intellectual tradition, yet modern. The book familiarizes students with the main tools and techniques they need to understand current research in formal semantics and contribute to the state of the art, and provides students with training in how to argue for one formalized theory over another on the basis of empirical evidence, through hypothesis comparison. We have used the book to teach a one-semester introduction to formal semantics for students who have already studied some linguistics, though no previous experience with logic is required. Beyond its use in traditional classroom settings, this book is suitable for flipped classrooms (i.e. classes where students read the textbook at home and use classroom time to ask questions and solve exercises) and for self-study.

One distinguishing feature of this book is the *Lambda Calcu-*

lator, an interactive, graphical application to help students practice derivations in the typed lambda calculus. It is designed for both students and teachers, with modules for online classroom instruction, graded homework assignments, and self-guided practice. The primary function is to assist in the computation of natural language denotations up a syntactic tree. To this end, the program detects common errors and attempts to provide intelligent feedback to the student user and a record of performance for the instructor. Many exercises in this textbook are designed to be solved with the Lambda Calculator. The software runs on Mac, Linux, and Windows machines. The student version of the calculator is available as a free download from `www.lambdacalculator.com`, which also provides documentation and exercise files; the teacher edition, which offers advanced functionality, is available to instructors on request by writing to `champollion@nyu.edu`. The Lambda Calculator was originally developed by Lucas Champollion, Maribel Romero, and Josh Tauberer (Champollion et al., 2007). Further contributions to its code and documentation have been made by Anna Alsop, Dylan Bumford, Oscar Dalton, Nigel Flower, Raef Khan, and Alex Warstadt, whose help we gratefully acknowledge. We are grateful to David Beaver, Simon Charlow, Dylan Chater, Tommy Grano, Adam Przepiórkowski, Gracie Rohde, and Ede Zimmermann for bringing non-trivial issues in previous versions to our attention.

Instructors who have previously taught from Heim & Kratzer (1998) will find much familiar material in this book, such as the composition rules: Function Application, Predicate Modification, Predicate Abstraction, and the Pronouns and Traces Rule. The most prominent difference in the framework is that we translate English expressions into well-formed expressions of the lambda calculus rather than specifying denotations directly using an informal metalanguage containing lambdas. Our style of analysis involves defining a formal *representation language*, which is a logic with a syntax and a semantics (the language of lambda calculus,

with some enhancements borrowed from the linguistic tradition), and defining a systematic *translation* from English to that language ('translate-first, interpret-second', in slogan form). Our logic-based representation language is both more precise and more compact than the informal language based on paraphrases adopted in Heim & Kratzer (1998). Our derivations easily fit into tree representations. Here is a sample derivation involving both Predicate Modification and Function Application:



Another important departure from the Heim & Kratzer (1998) framework is in the treatment of presupposition. Partial functions are replaced with total functions whose range includes an 'undefined' value, and a partiality operator is introduced. This means that Function Application is always defined, it is easy to read off

the presuppositions of a sentence from its logical translation, and definedness conditions do not get lost along the way.

There is also a greater emphasis on the notion of denotation relative to a model. This grounds our formal representation more firmly in intellectual tradition, and provides us with a method for capturing entailments, which we view as the primary source of data for a semantic theory.

We are grateful to Omar Agha, Masha Esipova, Alicia Parrish, Gracie Rohde, Aditya Yedetore, and Yuanyuan Zhang for assisting us with preparing the text of this book and for helping us create the answer keys. For helpful discussion, comments and feedback, we are grateful to Chris Barker, David Beaver, Brian Buccola, Ivano Caponigro, Simon Charlow, Dylan Charter, Natalie Clarius, Kathryn Davidson, Alexander Stewart Davies, Thomas Grano, Magda Kaufmann, Nathan Klinedinst, Karen Lewis, Dean McHugh, Adam Przepiórkowski, Zoltán Szabó, Guillaume Thomas, James Walsh, Thomas Ede Zimmermann, Joost Zwarts, and our semantics students at Boston University and NYU, respectively, from 2019 to 2025. Any remaining errors are of course our own.

Contents

1 Introduction

1.1 Compositional semantics

Natural languages allow us to communicate ideas of arbitrary complexity. As Chomsky (1965) put it in *Aspects of the Theory of Syntax* (p. 6), "an essential property of language is that it provides the means for expressing indefinitely many thoughts and for reacting appropriately in an indefinite range of new situations." Chomsky (1957, 1965) showed how RECURSION can be used to model some of the 'infinite generative capacity' of language. If a grammar is specified using recursive rules, then there are infinitely many strings that it can assign structural representations (parse trees) to (even if there are also infinitely many that it rules out).[1] This was an important step toward understanding the infinitude of language.

Recursion is just part of the story. What humans know when we know a language is not just which sentences are grammatical and which are ungrammatical and how to assign structural representations to strings of words (syntax), but also *what these sentences mean* (semantics). A corollary of our creative ability to express and understand indefinitely many new thoughts is that the

[1] To put it coarsely, a RECURSIVE system is one in which a concept or a procedure can be defined in terms of itself, while avoiding circularity. For example, the notion of 'well-formed sentence' can be defined recursively with recursive rules like this: If 'ϕ' is a well-formed sentence, and 'ψ' is a well-formed sentence, then 'ϕ and ψ' is a well-formed sentence. Chapter 3 will illustrate several instances of recursion, and we will continue to use it throughout the book.

process for assigning meanings to complex phrases must depend systematically on the meanings of the parts. In this sense, human knowledge of semantics is COMPOSITIONAL; in a compositional system of semantics, the meaning of a compound expression is a function of the meanings of its parts and the way they are syntactically combined (Partee, 1984, 281).

Important progress toward developing a theory of natural language's *semantic* infinitude was made around the turn of the 20th century by the logician/philosophers Gottlob Frege and Bertrand Russell. As depicted in the wonderful and highly-recommended graphic novel *Logicomix* (Doxiadis & Papadimitriou, 2009), Bertrand Russell's aim was to put mathematical reasoning on firmer logical footing, an aim that Frege was simultaneously and independently working towards. Their ideas nevertheless had powerful applications in natural language semantics. Mathematical and philosophical logic, championed also by Alfred Tarski, Rudolf Carnap, Ludwig Wittgenstein, Ruth Barcan Marcus, Donald Davidson, David Lewis, Alonzo Church, and Richard Montague, built up tools that suggested an answer to the question of how meanings are put together compositionally.

Following in that tradition, this textbook builds up several systems that, in a compositional manner, associate sentences with their TRUTH CONDITIONS. At some level, truth conditions are a way of characterizing the meaning of a sentence (although we will say more about 'meaning' below). Partee (2006) motivates this idea as follows:

> Knowing the meaning of a sentence does not require knowing whether the sentence is *in fact* true; it only requires being able to discriminate between situations in which the sentence is true and situations in which the sentence is false.

The truth conditions of a sentence determine the cases (or scenarios, or situations) under which the sentence is true. They don't

determine whether the sentence is in fact true, but they do determine what would have to be the case in order for the sentence to be true. TRUTH-CONDITIONAL SEMANTICS characterizes meaning by providing a systematic association between sentences and their truth conditions.

Logicians make use of truth conditions in order to give a theory of the conditions under which a proof is logically valid. The logician, mathematician, and philosopher Richard Montague suggested that the same technique can be applied to natural language. Two of his seminal works were *English as a Formal Language* (Montague, 1970) and *The Proper Treatment of Quantification in Ordinary English* (Montague, 1973b).[2] Barbara Partee, a linguist who championed and extended Montague's approach to natural language, illustrated in much greater detail how fruitful it could be through her work on all sorts of natural language phenomena. She, her many excellent students and grand-students, and many other linguists helped create the modern field of FORMAL SEMANTICS.

This book is an invitation to the field of formal semantics for students who may not have prior training in logic. The goal is for students to master a theory of semantic composition that has become a *lingua franca* in this field, one that Heim & Kratzer (1998) laid out in their important textbook, *Semantics in Generative Grammar*. It also builds heavily on an earlier textbook by Dowty et al. (1981), *Introduction to Montague Semantics*, which provides a pedagogical introduction to Montague's work on natural language semantics. With the help of recursive composition rules that assign meanings to complex phrases based on the meanings of the parts, the theory presented in this book makes it possible to assign truth conditions to infinitely many sentences in a compositional manner. Despite this infinitude, however, there is still a wide open world of interesting linguistic phenomena to explore under this

[2] Reprinted in a collection of his papers edited by Richmond H. Thomason as Montague 1974a and Montague 1974b respectively.

lens. Hence our invitation.

1.2 Implication relations

Let us stand back for a moment. This is a semantics book. Semantics is about meaning. What is meaning? It seems that meaning is somehow tied to understanding, insofar as understanding something amounts to grasping its meaning. So what is it to understand? For instance, does the Google search engine understand language? Many might argue that it does, at least to some extent. Case in point: at the time of writing, typing "350 USD in SEK" into a Google search returned "3062.33 Swedish Krona" as the first result. The ability to compute equivalences systematically and reliably is behavior that demands non-trivial depth of semantic processing.

How can we tell whether the computer really *understands*? One of the hallmarks of human understanding is the ability to draw INFERENCES; in other words, to understand something is to be able to determine its IMPLICATIONS. For example, at the time of writing, there was a web page that said:

(1) Natalie Portman speaks English and Hebrew fluently, and she also speaks Spanish, German, Japanese, and French.[3]

From this sentence, a reader would likely infer that Natalie Portman does not speak Russian—if she did, then Russian would have been listed among the languages she speaks. The sentence would not be *false*, strictly speaking, if it turned out that Natalie Portman also speaks Russian. Still, it *somehow* implies that she does not. A human or computer system that understands language would be able to draw this inference from the text. (Students who have studied semantics/pragmatics before will recognize this example

[3] `https://www.ranker.com/list/celebrities-who-are-bilingual/celebrity-lists`. Accessed August 20, 2019.

as a case of 'conversational implicature'; more on this notion below.)

Another inference that a reader would be licensed to draw is this:

(2) Natalie Portman speaks more than two languages.

This inference follows more strongly from (1): There's no way for (1) to be true without (2) being true as well. (In other words, this is a case of an 'entailment'; more on this notion below.) A system or agent that understands language—or grasps meaning—would be able to draw these kinds of inferences. In other words, a good theory of meaning should be able to explain when one sentence IMPLIES another sentence.[4]

We mean 'implies' here in a broad sense, one that covers several different specific types of implications. In this broad sense, our sentence (1) implies both:

- that Natalie Portman doesn't speak Russian, and
- that she speaks more than two languages.

These are not the same kind of implication, but they can both be classified under that broader umbrella.

One way of defining 'implies' in this broad sense is as follows: 'A implies B (in context C)' means: If someone says A (in context C), then a typical listener will conclude that B is true (assuming

[4]Terminological note: An IMPLICATION (or IMPLICATION RELATION) is a relation that holds between some sentences, called PREMISES, and another sentence, the CONCLUSION, when the conclusion follows from the premises. We say in that case that the premises IMPLY the conclusion. The noun *inference* normally describes the act of inferring conclusions from premises, but *inference* can also be used to mean *implication*. The verb *infer* is totally different from the verb *imply*, though; an intelligent person *infers* conclusions based on premises, but premises *imply* conclusions. The subject of *infer* is the person drawing the inference (the hearer). The subject of *imply* can either be the speaker, as in *John implied that he would be home late*, or the premise of an argument, as in *Sentence A implies Sentence B*.

that they trust the speaker). This notion covers a wide range of subtypes of implication, including entailments, implicatures, and presuppositions. Of these, entailments are of particular interest in formal semantics, so let us turn to these first.

1.2.1 Entailment

1.2.1.1 Entailment and reasoning

Entailment is a core notion in formal semantics, and it is closely connected with reasoning. Somebody who infers (2) *Natalie Portman speaks more than two languages* based on (1) *Natalie Portman speaks English and Hebrew fluently, and she also speaks Spanish, ...* reasons correctly. Somebody who infers based on

(3) Some lizards are pets.

that

(4) Some pets are lizards.

reasons correctly as well. But somebody who infers from

(5) All cats are animals.

that

(6) All animals are cats.

does not reason correctly. We will say that sentence (1) *entails* sentence (2), and that sentence (3) entails sentence (4). Sentence (5) does not entail sentence (6) under the definition of entailment that we will build our way up to in what follows.

Entailment can relate more than two sentences. For example, sentences (7a) and (7b) taken together entail (7c).

(7) a. Every man is mortal.
b. Socrates is a man.

c. ∴ Socrates is mortal.

The symbol ∴ in (7c) is pronounced "therefore". Similarly, in the following examples, the (a) and (b) sentences together entail the (c) sentence.

(8) a. If it rained last night, then the lawn is wet.
b. It rained last night.
c. ∴ The lawn is wet.

(9) a. Aristotle taught Alexander the Great.
b. Alexander the Great was a king.
c. ∴ Aristotle taught a king.

Each of these three sequences of sentences is an ARGUMENT, in the sense that it presents a CONCLUSION as a consequence of zero or more PREMISES.[5] In the case of (7), for example, the premises are (7a) and (7b), and the conclusion is (7c).

Arguments whose conclusion follows from their premises, like the ones in (7) to (9), are called VALID; others INVALID. In this book, we use the symbol ∴ for valid arguments and the symbol ∴̸ for invalid arguments. ENTAILMENT is defined as the relationship between the premise(s) and conclusion of a valid argument. So, if we have a theory of what makes a valid argument, we have a theory of entailment.

Validity is about reasoning correctly. What, then, is it to reason correctly or incorrectly? Consider again this invalid argument:

(10) a. All cats are animals. (Premise)
b. ∴̸ All animals are cats. (Conclusion)

The premise of this argument is true, but its conclusion is false.

[5] The term *argument* has other senses in addition to this one, as in for example *The couple had a huge argument yesterday, and nearly broke up* where *argument* means something like *verbal altercation*, or *The author's argument is that mass incarceration is an inevitable consequence of neo-liberal capitalism*, where the term *argument* is used as a synomym for *claim*.

Whenever that situation arises, an argument is invalid. But there are also *invalid* arguments with true premises and *true conclusions*:

(11) a. All cats are animals. (Premise 1: True)
b. Some animals are black. (Premise 2: True)
c. ∴ Some cats are black. (Conclusion: True)

Both premises of this argument are true, and so is its conclusion. But this is not correct reasoning. The conclusion doesn't *follow* from the premises. What exactly does this notion of *following* amount to?

While it's easy to see that reasoning from true premises to a false conclusion is not correct reasoning, it's much harder to put your finger on what goes wrong when we reason incorrectly from true premises to a true conclusion. What part of the reasoning is incorrect? That is, why is argument (11) not valid? Here is one way of thinking about it: It is not valid because one can find a counterexample. Imagine the argument is put to someone, Johnny, whose mastery of the English language is quite limited (think of a second language learner English at the beginner level, a small child, or a computer program, if you like). Johnny can understand some words like *all*, *are*, and *some*. He hasn't come across the words *animal*, *black*, and *cat* before, but he can tell that they are three different words (he likes to refer to them by their initial letters, *A*, *B*, and *C*). So to Johnny, the argument might as well look like this:

(12) a. All C are A. (Premise 1)
b. Some A are B. (Premise 2)
c. Some C are B. (Conclusion)

Let's say that even though Johnny doesn't understand the meanings of the words *A*, *B*, and *C*, he can still tell that each of them describes certain things in the world. And he can consider different possibilities or CASES. For example, it might be that A de-

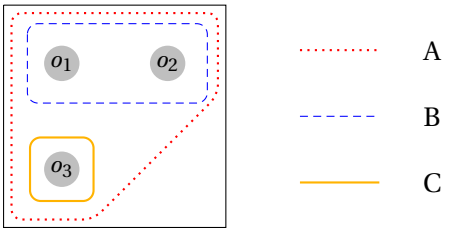


Figure 1.1: A case involving three objects (o_1, o_2 and o_3) where everything is A, and nothing is both B and C. In this case all C are A, and some A are B, so the two premises of (12) are true. But the conclusion ('Some C are B') is false. So this is a counterexample.

scribes everything there is, and that nothing is described both by B and by C, like in the case described by Figure 1.1. Notice that the words Johnny does understand—*all*, *are*, and *some*—keep the same meaning from one case to the next, while the words he does not understand are free to describe different things in different cases. Words of the first kind are called LOGICAL TERMS. Validity turns only on the logical terms, which is precisely why Johnny is in a position to judge it without knowing what *A*, *B*, and *C* mean.

Johnny thinks about this case and notices that he has found a COUNTEREXAMPLE to the argument: in this case, the two premises of the argument in (12) are both true but the conclusion is false. This means that the argument is not valid. If there is a hypothetical case where the premises are true and the conclusion is false, then the conclusion does not follow from the premises.

According to the way we use the term in this book, a CASE is something relative to which it makes sense to ask whether a sentence is true. For example, the sentence *All C are A* is true in the case depicted in Figure 1.1, but it is not true relative to some other cases. There are different ways to think about cases, which result in different notions of validity. One might think of them either as a 'model', 'structure', or 'interpretation' in the sense used in formal logic, or as what philosophers might call a 'possible world', an

'index' (a world/time pair), 'circumstance (of evaluation)', 'state of affairs', or 'scenario'. Without meaning to imply that all of the words just mentioned are synonyms, we have chosen the term CASE in order to remain neutral among these more specific notions as we give general definitions for the notions of validity and entailment.[6] Informally, cases can be thought of as the kinds of possibilities that Johnny considers, like the one illustrated in Figure 1.1.

With all this in mind, let us define validity as follows:

(13) An argument is VALID if and only if: In any case where all of the premises are true, the conclusion is true too.

We can determine whether the argument is valid by imagining all of the different sorts of cases that there are: Cases where nothing is *A*, and everything is both *B* and *C*, etc., etc. Among all of these different sorts of cases, is there one where the premises are true but the conclusion is false? If so, then the argument has a counterexample, and is therefore not valid.

Above, we defined entailment in terms of validity. Given the definition of validity that we now have, entailment between two sentences A and B can be defined as follows:

(14) A ENTAILS B if and only if:
In any case where A is true, B is true too.

[6] The notions of 'model' (or 'structure') in formal logic and 'possible world' in intensional semantics are related but distinct. For a purely extensional language like the one developed in Chapters 6–11, treating models as cases is "likely to be benign" (Partee, 1989, p. 94): variation across models can be understood either as variation in what the predicates mean or as variation in how the world is, and the two come apart only when we add intensional operators. Once we move to intensional semantics in Chapter 12, possible worlds go *inside* models rather than being identified with them — a model then consists of a *set* of possible worlds together with an accessibility relation, and it is the worlds within a fixed model that play the role of cases. For helpful discussion of the historical and conceptual relations among these notions, see Partee (1989).

Or in slogan form:
A ENTAILS B if and only if:
Whenever A is true, B is true too.

A theory of semantics should deliver predictions about when sentences in natural language stand in an entailment relation and when they do not; indeed, this is one of its most important jobs. The reader is encouraged to commit the slogan form of the definition in (14) to memory.

1.2.1.2 Valid vs. sound

As far as entailment and validity are concerned, it doesn't matter whether the premises are true in reality. These notions are not about evaluating truth in a single case; they are about what happens when you step back and consider every case. Indeed, an argument can involve correct reasoning and thus be valid even if it has false premises—in other words, even if the basis of the argument is not factual:

(15) a. Lemonade is made from watermelon. (False)
b. Watermelon is a type of vegetable. (False)
c. ∴ Lemonade is made from a type of vegetable. (False)

Of course lemonade is not actually made from watermelon, and watermelon is not actually a vegetable. But still the conclusion follows as an entailment from the premises. If we run it by Johnny, he will not be able to find a counterexample.

An argument that is valid and whose premises are furthermore actually true is called SOUND. So the argument in (15) is valid but not sound. Unlike validity, soundness depends on whether the premises are actually true, so knowing whether an argument is sound goes beyond Johnny's capabilities.

Both soundness and validity are useful concepts. In ordinary life, it matters a lot whether we reason from true or false premises. But it also matters whether we make mistakes in our reasoning

itself. Soundness is about correct reasoning from true premises, and validity is just about correct reasoning. The following argument is also valid but not sound. This argument has one false premise, one true premise and a true conclusion:

(16) a. Lemonade is made from watermelon. (False)
b. Watermelon is a type of fruit. (True)
c. ∴ Lemonade is made from a type of fruit. (True)

Exercise 1. Which, if any, of the following arguments are valid? Which, if any, are sound?

(i) Every Spaniard is female. The Statue of Liberty is a Spaniard. Therefore, the Statue of Liberty is female.

(ii) Every person is a person. Therefore, Paris is the capital of France.

(iii) There is no species of dog that is not a species of animal. The poodle is a species of dog. Therefore, the poodle is a species of animal.

Exercise 2. Can a valid argument have...

(a) false premises and a false conclusion?

(b) false premises and a true conclusion?

(c) true premises and a false conclusion?

(d) true premises and a true conclusion?

If you answer yes to any of these, give your own example of such an argument. If your answer is no, explain why.

1.2.1.3 Surface syntax as an unreliable guide to validity

An important observation about validity is that in order to determine whether a given argument in natural language is valid, one has to look deeper than the surface. Two arguments may be superficially similar, but differ in validity. For example, the following argument is valid:

(17) a. Lemonade contains **lemon**.
b. **Lemon** is a type of fruit.
c. ∴ Lemonade contains a type of fruit.

but the following is not:

(18) a. Lemonade contains **no vegetable**.
b. **No vegetable** is a type of mineral.
c. ∴ Lemonade contains a type of mineral.

Clearly, *no vegetable* has a very different kind of meaning from *lemon*. The former is a QUANTIFIER (in the sense that it is a noun phrase accompanied by a quantificational determiner like *no* or *every*), while the latter is a bare noun that stands for a kind. Although quantifiers and bare nouns can occupy (what seems superficially like) the same syntactic positions, they give rise to very different entailments.

Furthermore, because sentences in natural language can be AMBIGUOUS, the validity of an argument may depend on how the sentences in it are read. A word may have multiple different senses, of course; this is LEXICAL AMBIGUITY. There are also STRUCTURAL AMBIGUITIES. For example, consider the following argument. Is it valid?

(19) a. Some of Jane's emails are read.
b. Some of Jane's emails are unread.
c. ∴ Jane hasn't read one of her emails.

In one sense, yes, but in another sense, no. (19c) can be read either

as saying that it is not the case that there is an email of Jane's that she has read, or that there is an email of hers (a particular one) that she hasn't read. This difference is a SCOPE AMBIGUITY. On the first reading ("It is not the case that there is an email of hers that she has read"), the negation (*n't* in *hasn't*) TAKES SCOPE over the indefinite noun phrase *one of her emails*; on that reading the conclusion is incompatible with premise (19a), so the argument is not valid. On the second reading ("There is an email of hers that she hasn't read"), the indefinite noun phrase takes scope over the negation, and the conclusion follows from premise (19b). The formal tools that we will develop in this book will help to elucidate the various readings that scopally ambiguous sentences can have.

1.2.2 Implicature

Not every implication is an entailment. As mentioned above, there are several different sorts of implications, including both entailments and *implicatures*. Recall example (1), repeated here:

(20) Natalie Portman speaks English and Hebrew fluently, and she also speaks Spanish, German, Japanese, and French.

This sentence implies, in some sense of the word "implies", that Natalie Portman does not speak Russian. But suppose you observed Natalie Portman in a heated conversation with a Russian diplomat in perfectly fluent Russian. Would you conclude, based on this information, that (20) is *false*? Presumably not. So this implication is not an entailment. It derives from the assumption that the languages listed make up an exhaustive list of the languages that Natalie Portman speaks. If she did speak Russian and the author of sentence (20) knew this, they would be saying something misleading (or "lying by omission" as it's sometimes described colloquially, although arguably this is not a form of lying).

The implication that Natalie Portman doesn't speak Russian is an example of a CONVERSATIONAL IMPLICATURE (often referred

to in modern linguistics simply as an IMPLICATURE). Conversational implicatures are inferences that the hearer can derive using the assumption that the speaker is adhering to certain norms of conversation (Grice, 1975). Among these norms is the Maxim of Quantity, which requires that speakers provide as much information as needed for the information exchange (but not more). If we're on the subject of what languages Natalie Portman speaks, and if she speaks Russian, then the Maxim of Quantity dictates that this fact be mentioned.

Grice posits four maxims in total: Quantity (say as much as is required, but no more), Quality (do not say what you believe to be false, and have adequate evidence for what you say), Relation (be relevant), and Manner (avoid obscurity, avoid ambiguity, be brief, and be orderly). These four maxims all fall under what Grice calls the 'Cooperative Principle', which he sums up as follows: "Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of talk exchange in which you are engaged" (Grice, 1975, 45). Grice's work introduced the term 'conversational implicature' as a label for the kind of implication that arises through reasoning on the part of the hearer about the speaker's adherence to the Cooperative Principle and its constituent maxims.

The maxims interact as they give rise to implicatures. Whether or not a given sentence gives rise to a conversational implicature via the Maxim of Quantity depends on what is relevant, as the following exchange from the film *When Harry Met Sally* brings out:

> Jess: So you're saying she's not that attractive?
> Harry: No, I told you she is attractive.
> Jess: But you also said she had a good personality.
> Harry: She does have a good personality.
> Jess: When someone's not that attractive, they're always described as having a good personality.
> Harry: Look, if you would ask me, "What does she look like?" and I said, "She has a good personality."

> That means she's not attractive. But just because I happened to mention that she has a good personality, she could be either. She could be attractive with a good personality, or not attractive with a good personality.

Here, Harry is pointing out that the conversational implicature from *She has a good personality* to *She is not attractive* depends on what the QUESTION UNDER DISCUSSION (the subject matter at hand) is. Hence the Maxim of Relation interacts with the Maxim of Quantity.

Exercise 3. What is the relation between the terms *implication* and *implicature?* Explain using definitions and at least one example of each.

Conversational implicatures differ from entailments in the following way: Suppose that A is true. If A entails B, then B is true for sure, but if A conversationally implicates B, then B is not guaranteed to be true. Implicatures can be CANCELLED without producing a contradiction. Recall that our Natalie Portman sentence (20) implies (21):

(21) Natalie Portman does not speak Russian.

As a conversational implicature, this inference is DEFEASIBLE: it is possible to assert (20) and deny (21) without contradiction. For example, one could say, without contradicting oneself:

(22) Natalie Portman speaks English, Hebrew, Spanish, German, Japanese, and French. In fact, she speaks Russian as well.

Here the implicature is being CANCELLED (or DEFEATED). The second sentence of (22) negates (21), the implicature of (20) (since

the implicature was itself negative: that Natalie Portman does not speak Russian). Nothing goes wrong here; if your friend made these two claims in succession, you could not accuse her of contradicting herself.

In contrast to conversational implicatures, entailments are not defeasible. Observe:

(23) Natalie Portman speaks English, Hebrew, Spanish, German, Japanese, and French. #In fact, she doesn't speak more than two languages.

(The hash-mark # here indicates that the sentence is somehow odd in its interpretation. In general, the hash-mark is used to indicate that a sentence is either semantically anomalous—makes no sense—or pragmatically infelicitous, i.e., inappropriate, in a given context. This symbol is the semantics/pragmatics equivalent of the asterisk [*] used in the syntax literature to indicate that a sentence is ungrammatical.) If your friend uttered these two sentences in succession, she would be open to the accusation that she was contradicting herself. Hence this implication is an entailment.

This DEFEASIBILITY TEST is a way to test whether the implication from a sentence A to sentence B is an entailment or an implicature. To run the defeasibility test for an implication from sentence A to sentence B, the first step is to construct a text of the form A & *not-B*, where *not-B* negates B, and & is the most appropriate conjunction (*but*, *and*, or *and in fact*, whichever fits best). Then ask a native speaker of the language in question about whether the constructed example is contradictory. If so, then the implication is not defeasible, which suggests that it is an entailment rather than an implicature.[7]

[7] There are defeasible inferences that are not implicatures. Among these are inferences based on real-world knowledge. For example, *John smokes* loosely implies *John buys cigarettes* – if John smokes, then he probably buys cigarettes, but it's possible that he doesn't, so the inference is defeasible. This case is not

Let's consider another example. Does (24a) entail (24b)?

(24) a. Some Republicans voted 'yes'.
b. Not all Republicans voted 'yes'.

To run the defeasibility test on this example, it is necessary to construct a sentence of the form *A & not-B*. For this example, to form *not-B*, the negated version of (24b), it suffices to take away the *not* preceding *all*, as in *All Republicans voted 'yes'*.

(25) Some Republicans voted 'yes'; in fact, all of them did.

Now we can ask whether *A & not-B* is self-contradictory. Would someone who uttered (25) be contradicting themselves? No. So *A & not-B* is not self-contradictory in this case. So the implication from (24a) to (24b) is not an entailment; it's a conversational implicature. This particular example is called a SCALAR IMPLICATURE, a type of implicature based on the maxim of quantity that arise in the context of a scale of alternatives—in this case *some* and *all*—arranged from weakest to strongest.

1.2.3 Presupposition

In Chapter 8, we will see another type of implication, besides entailment and implicature, called PRESUPPOSITION. When one sentence presupposes another, the other is treated as background information, or in other words, taken for granted. For example, *Sue stopped smoking* presupposes that Sue smoked in the past. Presuppositions are generally considered non-defeasible just like entailments, but they have some distinctive properties that set them aside from ordinary entailments. Chief among them is that they PROJECT OVER NEGATION; for example, *Sue didn't stop smoking* implies that Sue smoked in the past just as its affirmative counterpart does. The PROJECTION TEST uses this property: if a sentence *A* and

an implicature, because it's not an inference that crucially relies on reasoning about the speaker's adherence to norms of conversation.

its negation both entail another sentence B, then B is a presupposition of A. More on this in Chapter 8.

1.2.4 Semantics vs. pragmatics

This section has introduced three different types of implications: entailments, implicatures, and presuppositions. These lie to varying extents in the purview of semantic theory. Semantics is sometimes said to be the study of what *linguistic expressions* mean, while pragmatics is the study of what *speakers* mean by them. (By LINGUISTIC EXPRESSIONS, we mean to include words, phrases, and sentences—any chunk of language that forms a syntactic unit.) The term 'pragmatics' can also be applied to the study of any interaction between meaning and context, broadly construed. There is no sharp dividing line between semantics and pragmatics, and indeed the study of presupposition arguably lies within their intersection. However, it is fair to say that implicatures lie in the domain of pragmatics proper, while entailments lie properly in the domain of semantics. However one defines 'semantics', it is undeniable that truth conditions are part of it, and entailment is a relation between sentences that is grounded in truth conditions.

A decision procedure for distinguishing between the various types of implication relations is summarized in Figure 1.2. Use the defeasibility test to distinguish between entailment and implicature; use the projection test to distinguish between ordinary entailment and presupposition.

Exercise 4. Consider the following pairs of sentences:

(i) a. Every dog barked.
    b. Every small dog barked.

(ii) a. Every dog made a small noise.
    b. Every dog made a noise.

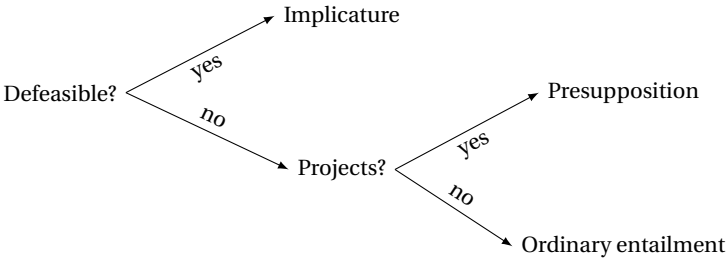


Figure 1.2: A decision tree for categorizing implications

(iii) a. Sam regrets winking at Dave.
b. Sam winked at Dave.

(iv) a. Sam lived in London in the 1990s.
b. Sam doesn't live in London now.

(v) a. When I was in the army, I tried LSD.
b. I was in the army.

(vi) a. It's warm.
b. It's not hot.

In each case, the first implies the second. Are these implications entailments? Provide empirical support for your answers using the defeasibility test.

Note: In some cases, (a) presupposes (b). Presupposition can be considered a species of entailment, so even if you think that the example is a presupposition, it still makes sense to conclude on the basis of a defeasibility test that it is an entailment (some sort of entailment, even if not an ordinary one).

1.3 Semantic relations

1.3.1 Semantic relations beyond entailment

A truth-conditional semantics delivers predictions not only about entailment, but also other semantic relations. Entailment is a SEMANTIC RELATION in the sense that it is grounded in *truth*: A entails B if and only if *whenever A is true, B is true too.* Implicature is not a semantic relation because it is grounded in norms of communication, which go beyond truth.

EQUIVALENCE is another semantic relation. Two sentences are EQUIVALENT if they are mutually entailing; whenever one is true, the other is true, and vice versa. For example, the following pairs of sentences are equivalent:

(26) a. I saw neither the antelope nor the snake.
b. I didn't see the antelope and I didn't see the snake.

(27) a. Sue is Mary's sibling.
b. Mary is Sue's sibling.

(28) a. John is in front of Bill.
b. Bill is behind John.

There are no cases where the (a) sentences among these are false and the (b) sentences are not, or vice versa. Strictly speaking, though, that holds only for cases that keep the meanings of the individual words fixed. The first pair is guaranteed by the meanings of *neither... nor* and *not... and not...* alone, which are logical terms of the sort Johnny understands. The other two depend on facts about particular words: that *sibling* describes a symmetric relation, and that *in front of* and *behind* describe converse ones. Entailments of this second kind can be built into a semantic theory by stating them explicitly as *meaning postulates*, a device we return to in Chapter 7.

The semantic relations also include two different types of opposition that sentences can stand in to each other. For instance,

consider the following sentence:

(29) Everybody likes chocolate.

To deny this statement, you might say:

(30) Not everybody likes chocolate.

Whenever (29) is true, (30) is false, and vice versa. Together, they "divide the true and false among them" (cf. Horn 2018). In this sense, (29) and (30) stand in CONTRADICTORY OPPOSITION.

A stronger denial of (29) would come from the following:

(31) Nobody likes chocolate.

Certainly this sentence couldn't be true simultaneously with (29), but it's an extreme statement, so both could be false; maybe some like chocolate and some do not. Sentences (29) and (31) stand not in contradictory opposition, but rather CONTRARY opposition.

What contrary and contradictory opposition have in common is that the two sentences are INCOMPATIBLE, or MUTUALLY INCONSISTENT. That is, they cannot be true at the same time. Where they differ is in whether they are MUTUALLY EXHAUSTIVE, i.e., whether there are cases that are left out by both. These two sorts of opposition can be defined as follows:

(32) A sentence A stands in CONTRADICTORY OPPOSITION to a sentence B if and only if:
It is impossible for A and B to be true together, and it is impossible for A and B to be false together.

(33) A sentence A stands in CONTRARY OPPOSITION to a sentence B if and only if:
It is impossible for A and B to be true together, but it is possible for A and B to be false together.

Contrary opposition generally involves two extremes, so that there is a middle ground, where neither hold.

Exercise 5. For each of the following sentences, give (a) a sentence that stands in contrary opposition to it and (b) a sentence that stands in contradictory opposition to it.

(i) My pet giraffe is young.

(ii) I always drink coffee in the morning

(iii) The evidence proves that he is guilty.

(iv) Everyone liked it.

1.3.2 Square of opposition

A famous constellation of semantic relations is the SQUARE OF OPPOSITION, shown in Figure 1.3. The square of opposition has four corners, A, I, E, and O, which come from the vowels in the Latin words *affirmo* and *nego*. The affirmative side is the lefthand side, with *Every sailor is a pirate* and *Some sailor is a pirate*, a universal and a particular (a.k.a. existential) statement, the former above the latter in the square. On the righthand, negative side, we have *Every sailor is **not** a pirate* and *Some sailor is **not** a pirate*, again a universal and an existential. This visualization was created by medieval scholars who were studying the work of Aristotle. Aristotle's *De Interpretatione*, among many other ancient works, contributed insights to what we now call natural language semantics that are still relevant today, and ancient and medieval scholars such as Boethius and Buridan built on that work, long before Frege and Russell.

Now along with the sentences at the corners, the lines connecting these sentences are labelled with various semantic relations. In the medieval depiction of the square of opposition, the diagonal lines are labelled 'contradictory' (in Latin), signifying that

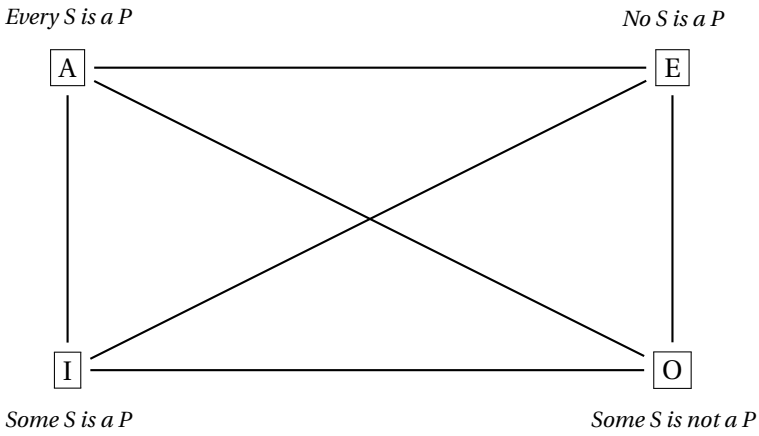


Figure 1.3: The square of opposition, with the lines connecting the corners left unlabelled. Shorthand: *S* = *sailor*; *P* = *pirate*.

the two pairs of sentences that stand on opposite corners of the square stand in contradictory opposition. The two sentences at the top are labeled as standing in contrary opposition.

The square of opposition thus makes a number of claims involving semantic relations, including:

- Across the diagonal, A *Every S is a P* and O *Some S is not a P* stand in contradictory opposition.
- Across the other diagonal, E *No S is a P* and I *Some S is a P* stand in contradictory opposition.
- At the top, A *Every S is a P* and E *No S is a P* stand in contrary opposition.

If two sentences stand in contradictory opposition, then there is no case in which they are both true, and no case in which they are both false. Some cases to consider are illustrated in Table 1.1. In case (i), for example, where a and c are the sailors and a and b are the pirates, *Every S is a P* is false and *Some S is not a P* is

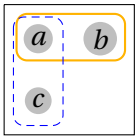
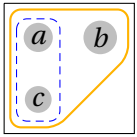
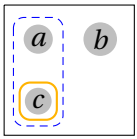
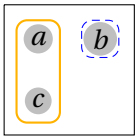
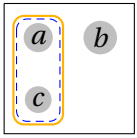
i.	sailors: a, c	pirates: a, b	
ii.	sailors: a, c	pirates: a, b, c	
iii.	sailors: a,c	pirates: c	
iv.	sailors: b	pirates: a,c	
v.	sailors: a,c	pirates: a,c	

Table 1.1: Some different scenarios

true. In case (ii), where the sailors are a and b and the pirates are a, b, and c, *Every S is a P* is true and *Some S is not a P* is false. The reader is invited to inspect the remaining cases, and verify that the truth values of the two sentences are always opposite. This is as expected under the view that these two sentences stand in contradictory opposition. The same holds for the E and I corners.

If two sentences can be true simultaneously then they do not stand in contradictory or contrary opposition. For instance, at the bottom, I *Some S is a P* and O *Some S is not a P* do not stand

in contrary or contradictory opposition, because they could both both true. For instance, in case (i) in Table 1.1, *Some S is a P* is true, and *Some S is not a P* is also true. The fact that these two sentences are mutually consistent is reinforced by the fact that one would not be guilty of self-contradiction for asserting, *Some sailors are pirates and some sailors are not.*

Exercise 6. For each corner of the square of opposition (A, I, E, O), identify one case in Table 1.1 where the corresponding sentence is true, and one case where it is false. You can identify the case by its lowercase Roman numeral (i, ii, iii, iv, v).

Exercise 7. Match the **definition** to the correct **semantic relation**.

Semantic relations:

(a) A and B are equivalent.

(b) A and B are incompatible.

(c) A entails B.

(d) B entails A.

Definitions:

(i) In every case where A is true, B is true too.

(ii) In every case where B is true, A is true too.

(iii) A entails B and B entails A.

(iv) There is no case where A and B are true at the same time.

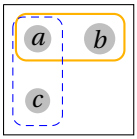
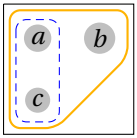
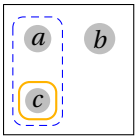
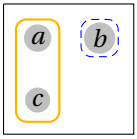
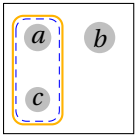
		A. At least one sailor is a pirate.	B. At least one pirate is a sailor.	C. At least one sailor is not a pirate.	D. No sailor is a pirate.	E. Every sailor is a pirate.	F. Every pirate is a sailor.	G. Every sailor is a non-pirate.	H. Not every sailor is a pirate.	I. It's not true that no sailor is a pirate.
i.		T	T	T					T	T
ii.		T	T			T				T
iii.		T	T	T			T		T	T
iv.				T	T			T	T	
v.		T	T			T	T			T

Table 1.2: Truth-value judgments for sentences in cases

Exercise 8. Assume the truth value judgments in Table 1.2. A 'T' under a sentence in the row corresponding to a given image means that the sentence is true relative to the scenario depicted in the image.

Notice that some pairs of sentences are true in exactly the same set of cases (among the five possible cases considered here). Recall that sentences that are true in exactly the same set of cases are called **equivalent**.

For each of the following sentences (i.-iii.), specify an equivalent sentence from among the given alternatives by choosing the lettered sentence (a-d).

Sentences:

(i) At least one sailor is a pirate.

(ii) At least one sailor is not a pirate.

(iii) No sailor is a pirate.

Choose an equivalent from among:

(a) Every sailor is a non-pirate.

(b) Every sailor is a pirate.

(c) At least one pirate is a sailor.

(d) Not every sailor is a pirate.

Exercise 9. For each of the long and complicated sentences (a)-(c), choose a logically equivalent way of rewriting it (a way of rewriting it which does not affect which cases the sentence is true in) from among (i)-(iv). Hint: use the square of opposition.

Long and complicated sentences:

(a) It is not the case that at least one pirate is a sailor.

(b) It is not the case that at least one sailor is not a pirate.

(c) It is not the case that no sailor is a pirate.

Choose from the following simpler options:

(i) Every sailor is a pirate.

(ii) No pirate is a sailor.

(iii) At least one sailor is a pirate.

(iv) At least one sailor is not a pirate.

To summarize, this section has presented a series of semantic relations, including entailment, equivalence, and contradictory and contrary opposition. What all of these semantic relations have in common is that they are grounded in truth conditions. A truth-conditional semantics for some fragment of English determines the semantic relations that sentences of that fragment stand in to each other: which sentences entail which other sentences, which sentences stands in contradictory opposition, etc. One aspect of the square we have set aside is what happens in cases where the restrictor noun has an empty extension — for example, when there are no sailors at all. Whether *Every sailor is a pirate* is true, false, or something else in such a situation turns out to be a subtle question, and answering it properly requires the theory of PRESUPPOSITION developed in Chapter 8.

1.4 Style: Indirect interpretation

This book presents the reader with a system for assigning truth conditions to sentences of natural languages (mostly English) in a compositional manner. We will go about this with the help of an unambiguous formal language for representing truth conditions. Our formal language will be a LOGIC, roughly, a formal language in which it is clearly defined which arguments are valid and which are not. (We will develop different formal languages as we go along, but a given semantic system only uses one at a time.) We refer to the formal language that we use for representing meanings as a REPRESENTATION LANGUAGE.

The representation language serves as an intermediary between the natural language of interest and its semantics. In this book, we define systems that systematically *translate* expressions of natural language (e.g. English) into the representation language. The representation language expressions have specified meanings, and we let the natural language expressions inherit their meanings from their representation-language translations. This style of analysis is called INDIRECT INTERPRETATION. Indirect interpretation is the method used in Richard Montague's paper, 'The Proper Treatment of Quantification in Ordinary English' (Montague, 1974b).

Another important concept in the mix here is META-LANGUAGE, which is typically contrasted with OBJECT LANGUAGE. In discourse about language, the object language is the language that is being talked *about*, while the language *in which* the talking occurs is the meta-language. (The related term META-LINGUISTIC is used to describe discourse that is about language.) In this book, we are using English (plus some talk of sets) to theorize about English, so the object language is English, and so is the meta-language (with some mathematical notions mixed in). We will also use it to talk about the formal representation languages introduced later in the book, which are object languages in their own right; we return to this point below. If we were developing a theory of French, then French would be the object language, and we might still use En-

glish to talk about it. If this book were translated into Spanish, then Spanish would be the meta-language, even if the example sentences were left in English.

Another way to go about things would be to skip the logic and give the interpretations of object language expressions directly using our meta-language, as Richard Montague did in his paper 'English as a Formal Language' (Montague, 1974a). That style is known as DIRECT INTERPRETATION, and it is adopted in the Heim & Kratzer (1998) textbook.

The indirect interpretation style offers a number of practical technical advantages over direct interpretation. One advantage derives from the fact that natural language is ambiguous, while our logical representation language is not. Having a non-ambiguous representation language makes it possible to make precise predictions about entailment and other semantic relations like contradictory vs. contrary opposition and equivalence. A more practical advantage is that it allows our meaning representations to be more concise, so they can fit on a tree diagram showing the compositional derivation of the meaning of a sentence. It also meshes well with the Lambda Calculator, a pedagogical software application that is integrated with this book.

Exercise 10. Underline the object-language expressions in the following meta-linguistic statements.

(a) The word *boy* contains three letters.

(b) John said, "I am hungry."

(c) John said that he was hungry using the word *hungry*.

(d) The English first person pronoun rhymes with *eye*.

Part of our task as theorists is to specify the syntactic and se-

mantic rules for our representation language. When we are laying out the rules of a formal logic, we are again talking about a language, albeit a formal language. In that setting too, there is an object language and a meta-language; it's just that the object language happens to be a formal logic. So in the picture we build up in this book, there will actually be two languages that play the role of 'object language': the natural language whose semantics we aim to characterize (a fragment of English), and the formal language with which we represent the meaning, i.e., the 'representation language'. To avoid confusion, we will refer to these as the 'natural language' and the 'representation language', respectively, rather than as the 'object language'.

Exercise 11. What is the difference between direct and indirect interpretation? Which style is used in this book?

1.5 Limitations of truth-conditional semantics

It is sometimes suggested that truth conditions are *all there is* to the meaning of a sentence. Wittgenstein writes in his *Tractatus*: "To understand a sentence means to know what is the case if it is true."[8] Heim & Kratzer (1998) begin similarly: "To know the meaning of a sentence is to know its truth conditions." In this spirit, one might assume that the meaning of a sentence consists *entirely* in its truth conditions.

There is certainly more to meaning, though. In the two quotations above, truth conditions are associated with sentences, rather than with particular occasions on which these sentences are used. A SENTENCE is a particular word sequence that could in principle be used on many different occasions, or on none; an UTTERANCE on the other hand is a sentence as produced on a given occasion.

[8] Wittgenstein (1921) 4.024; our translation.

An utterance is typically associated with a designated speaker, addressee, time, and location, but a sentence is not. An utterance is also situated in a particular discourse context, where some things are relevant and under discussion and other things are not. It is useful to distinguish accordingly between SENTENCE MEANING and UTTERANCE MEANING. The implicatures that an utterance gives rise to in its context can be seen as part of its meaning. Utterances have meaning beyond truth conditions.

Sentence meaning goes beyond truth conditions too. In fact, it is not clear that all sentences even have truth conditions. Declarative statements of opinion such as *Vegemite is tasty*, commands like *Eat your vegemite!* and questions like *Did you eat your vegemite?* are among the types of sentences that have been argued not to have truth conditions, although opinions vary on these issues. We focus here on sentences that do—declarative statements of fact like *Vegemite consists mainly of brewer's yeast extract.* The techniques we will develop for this purpose can profitably be extended to a wider range of sentence types once they are in place.

But even the meaning of declaratives goes beyond truth conditions. For one example, consider *Sue has a twin* vs. *Sue is a twin* (example due to Matt Mandelkern). These two sentences have the same truth conditions, but differ in how easy they make it to refer to Sue's twin with a pronoun in the next sentence.

(34) a. Sue has a twin. She's at boarding school.
b. Sue is a twin. She's at boarding school.

In (34a), the pronoun *she* is most naturally interpreted as referring to Sue's twin. In (34b), it has to refer to Sue.

The earliest well-known example of this kind is due to Barbara Partee (cited in Heim 1982):

(35) a. I dropped ten marbles and found nine of them. ?It is probably under the sofa.
b. I dropped ten marbles and found all of them, except

one. It is probably under the sofa.

The question mark indicates that the sentence it precedes is degraded — marginal, but not sharply unacceptable. DYNAMIC SEMANTICS models this contrast as a difference in meaning, but this is beyond the scope of this book. Our system will be STATIC.

One final limitation of truth-conditional semantics that we will mention here is that truth conditional meaning is somewhat coarse-grained, collapsing finer-grained distinctions making up a phenomenon known as HYPERINTENSIONALITY (e.g. Muskens 2005). For example, any two sentences expressing mathematical truths ($2+2=4$ and $e^{i\pi}=-1$) have the same truth conditions—they're true in all cases—but they have different meanings. Here is an argument for the claim that they have different meanings: The sentence 'Ed knows that $2+2=4$' doesn't entail 'Ed knows that $e^{i\pi}=-1$'; therefore, $2+2=4$ must mean something different from $e^{i\pi}=-1$. We won't have much to say about hyperintensionality in this book, but we acknowledge that it is an important dimension of meaning.

Exercise 12. What is truth-conditional semantics?

Exercise 13. In what ways does meaning go beyond truth conditions? List three and explain each.

Welcome!

Consider this book a starter kit for a theory of semantics. If you understand the foundations well, you will be able to modify them to suit your purposes. In trying to extend the theory to account for a certain phenomenon, you may well find yourself making a

fundamental contribution to the theory of natural language semantics.

2 Sets, relations, and functions

2.1 Introduction

Remember the square of opposition, with its four corners. The left-hand side, the affirmative side, furnishes an affirmative universal A and an affirmative existential I, for example:

A *Every sailor is a pirate.*

I *Some sailor is a pirate.*

And on the right-hand side, the negative side, there is a negative universal E and a negative existential O, for example:

E *No sailor is a pirate.*

O *Some sailor is not a pirate.*

In the terminology of Anderson (2019), E expresses FULL NEGATION (all are not) and O expresses PARTIAL NEGATION (some are not).

Of course, Aristotle did not talk about English sentences but about ancient Greek ones, but the distinctions he drew are language-independent. In modern formal semantics and philosophy of language, it is common to distinguish between a sentence and a PROPOSITION. While a sentence is a grammatical string of words in a particular language, a proposition is a language-independent entity. In general, a proposition is the abstract content or meaning

expressed by a declarative sentence, independent of the specific words or language used to convey it. For example, the English sentence *Every sailor is a pirate* and its Ancient Greek translation are different sentences, but they express the same proposition.

Each of the corners of the square carries a sentence that expresses what Aristotle called a CATEGORICAL PROPOSITION, in Aristotle's terms (a proposition involving categories, like 'sailor' and 'pirate', as opposed to particular individuals or entities like Socrates). The proposition expresses a relation between categories.

The relations among categories that are signified by categorical propositions can be visualized using circles. You have probably heard of Venn diagrams; here we are using Euler diagrams, another visualization technique in this category. The circles, of course, signify the categories, and the visual relation among the circles reflects the abstract relation among categories. For example, to say *Every S is a P* is to place the S's among the P's; so, the P-circle encompasses the S-circle.

The visualizations in Figure 2.1 display relations among sets that can be expressed in the language of set theory, and they suggest the following hypotheses about the meanings of the logical words that figure in these sentences, *every*, *some*, *no*, and *not* (the highlighted terms will be defined subsequently):

- ***Every* as subset**
  The word *every* can be thought of as expressing that one set is a SUBSET of the other.

- ***Some* as non-empty intersection**
  The English determiner *some* can be thought of in terms of INTERSECTION, as a relation between two sets X and Y which holds if and only if there is some member of X which is also a member of Y, i.e., if and only if the intersection between X and Y is NON-EMPTY. For instance, *some sailor is a pirate* is true if and only if there is some individual which is both a sailor and a pirate.

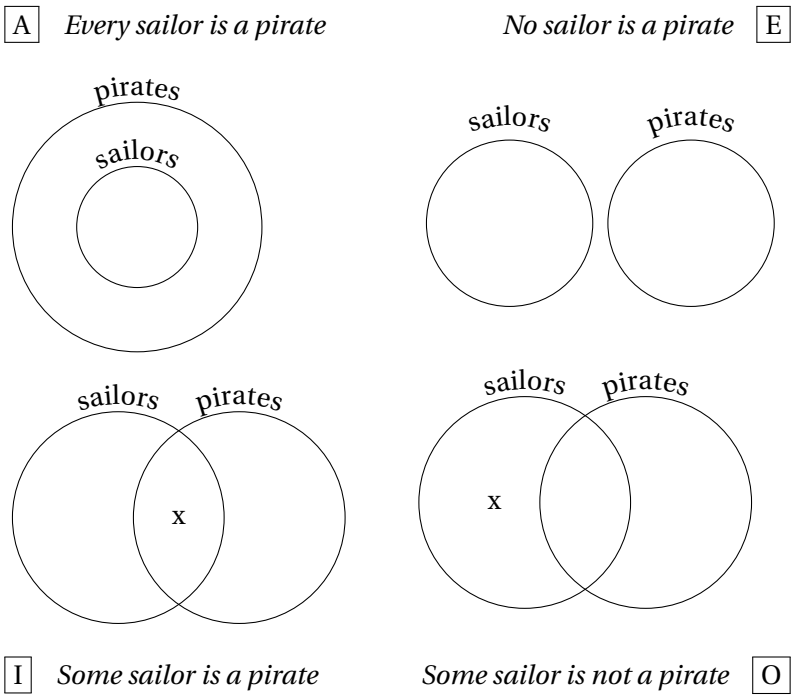


Figure 2.1: Euler diagrams for the square of opposition

- ***No* as empty intersection**
  The determiner *no* can be thought of as a relation between two sets X and Y which holds if the two sets have no members in common, in other words, if and only if they have an EMPTY INTERSECTION. So *no sailor is a pirate* holds if and only if there is no individual who is both a sailor and a pirate.

- ***Not* as set complement.** The verbal negation that occurs in O *Some sailor is not a pirate* can be characterized in terms of SET COMPLEMENT: the result of 'subtracting' one set from another.

The next section presents definitions of the set-theoretic terms that have appeared in this list, along with related terms, and some notation from set theory. In doing so, this chapter thus builds up a core part of our theory: the meta-language. Recall that truth conditions boil down to a specification of the sorts of cases relative to which the sentence would be judged true or false. In this book, following the tradition in mathematical logic, we assume that these *cases comprise sets and relations*. The symbols of set theory that we present here will thus form part of our meta-language for describing the sorts of cases that a sentence can be true or false relative to.

2.2 Set theory concepts

2.2.1 What are sets?

Set and elements. A SET is an abstract collection of distinct objects, which are called the MEMBERS or ELEMENTS of that set.[1] Here is an example of a set:

$$\{2, 7, 10\}$$

This set contains three elements: the number 2, the number 7, and the number 10. The members of the set are separated by commas and enclosed by curly braces. To express the fact that 2 is a member of this set, we write:

$$2 \in \{2, 7, 10\}$$

This expression is a declarative statement, which can be read aloud as follows: "2 is a member of the set containing 2, 7 and 10." The symbol $\in$ is pronounced "is an element of" or "is a member of" or

[1] The material in this section is inspired heavily by Partee et al. (1990), where you will find an excellent and a more in-depth presentation of these and related issues.

"is in". To express the fact that 3 is *not* a member of this set, we write:

$$3 \notin \{2,7,10\}$$

This statement can be read, "3 is not a member of the set containing 2, 7 and 10."

Unordered. The elements of a set are not ordered. Thus this set:

$$\{2,5,7,4\}$$

is exactly the same set as this set:

$$\{5,2,4,7\}$$

Likewise, listing an element multiple times does not change the membership of the set. Thus:

$$\{3,3,3,3,3\}$$

is exactly the same set as this one:

$$\{3\}$$

Equal sets. Two sets are EQUAL (alternatively, IDENTICAL) if and only if they have the same members. It follows that there is only one set containing 1, 2, and 3 and nothing else; the expressions $\{1,2,3\}$, $\{1,3,2\}$, and $\{3,2,1\}$ are simply three ways of writing that one set. Fact: given two sets A and B, $A = B$ if and only if both $A \subseteq B$ and $A \supseteq B$, where $\subseteq$ is the subset relation and $\supseteq$ the superset relation, both defined below.

Sets can be infinite. Here is another example of a set:

$$\{2,4,6,8,\ldots\}$$

The ELLIPSIS NOTATION $(\ldots)$ signals that the list of elements continues according to the pattern. So this set is infinite; it contains all positive even numbers.

Elements can be concrete or abstract. In the kind of set theory that linguists typically use, elements may be either concrete (like the beige 1992 Toyota Corolla the first author sold in 2008, you, or your computer) or abstract (like the number 2, the English phoneme /p/, or the set of all professional soccer players of the 1980s).

Known or unknown. Partee et al. (1990) also point out:

> A set may be a legitimate object even when our knowledge of its membership is uncertain or incomplete. The set of Roman emperors is well-defined even though its membership is not widely known ..., although it may be hard to find out who belongs to it. For a set to be well-defined it must be clear *in principle* what makes an object qualify as a member of it ...

By contrast, the set of all *good* Roman emperors is not well-defined in this sense. Reasonable people disagree about what makes a ruler good, so there is no criterion that settles, even in principle, which emperors would belong to it.

Predicate notation. When we can't list all of the members of a set, we can use PREDICATE NOTATION (also called SET-BUILDER NOTATION) to describe the set of things meeting a certain condition. To do that, we place a VARIABLE – a symbol that serves as a placeholder – on the left-hand side of a vertical bar, and put a description containing the variable on the right-hand side. In principle, we are free to choose any symbol we like to serve as a variable, but typical choices for numbers are single letters in the middle of the alphabet like n, m, and k. Let us use n as our variable. For example, the following expression describes the set of integers below zero ($\mathbb{Z}$ designates the set of integers):

$$\{n \mid n \in \mathbb{Z} \text{ and } n < 0\}$$

This expression can be read, 'the set of all n such that n is an integer and n is less than 0'. The vertical bar can be read as 'such that' in this context; it is sometimes written as a colon. The same set could be written as

$$\{-1,-2,-3,\ldots\}$$

showing that the set of elements stretches out infinitely in the negative direction.

Singleton sets. A set need not have multiple members; it can have just one element:

$$\{3\}$$

This set contains just the number 3. If a set has only one member, it is called a SINGLETON; we also say that the set $\{3\}$ is the singleton of 3. A set can even be empty.

Empty set. The set that contains no elements is called the EMPTY SET, and it can be written either as $\{\}$ or, more conventionally, as $\varnothing$.

Cardinality. The CARDINALITY of a set is the number of elements it contains. The cardinality of the empty set, for example, is 0. Cardinality is expressed by vertical bars surrounding the set: If A is a set, then $|A|$ is the cardinality of A. So, for example:

$$|\{5,6,7\}| = 3$$

This formula can be read, 'The cardinality of the set containing 5, 6, and 7 is 3.'

Singleton. A SINGLETON SET is a set containing only one element (with a cardinality of one).

Exercise 1. What is the cardinality of the following sets?

(a) $\{2,3,\{4,5,6\}\}$

(b) $\varnothing$

(c) $\{\varnothing\}$

(d) $\{\varnothing,\{3,4,5\}\}$

(e) $\{\varnothing,3,\{4,5\}\}$

Sets containing sets. The members of a set can be all sorts of things. A set can even contain *another set* as an element. The following set:

$$\{2,\{1,3,5\}\}$$

contains *two elements*, not four. One of the elements is the number 2. The other element is a three-membered set. A set could also, of course, contain the empty set as an element, as the following set does:

$$\{\varnothing,2\}$$

This set has two elements, not one.

The only kinds of things that cannot be members of a given set are certain collections that cannot count as sets at all (technically, *proper classes*). This restriction is needed in order to avoid problems such as RUSSELL'S PARADOX (there cannot be a set of all sets that are not members of themselves, since that set could neither contain nor fail to contain itself). For now, we will set this paradox aside. In Chapter 5, we will introduce a simplified version of Russell's solution to his paradox called *type theory* that constrains the conditions under which one set can be a member of another.

2.2.2 Relations among sets

Subset. The top-left corner of Figure 2.1 gives a visual depiction of A *Every sailor is a pirate* with one circle nested in another. This diagram is a visualization of the notion of subset. Subset is a relation that holds between two sets, defined as follows: A set A is a SUBSET of a set B, written

$$A \subseteq B$$

if there is no member of A that is not also a member of B. Hence for any given object x, if $x \in A$ then $x \in B$. The symbol $\subseteq$ is pronounced "is a subset of".

The set $\{a, b\}$ is a subset of $\{a, b, c\}$. The latter is not a subset of the former, because the latter contains a member that is not a member of the former, namely c. But $\{a, b\}$ is a subset of $\{a, b, c\}$, because there is no member of the former that is not also a member of the latter.

Empty set as subset of every set. The empty set is a subset (not an element!) of every set. So, in particular:

$$\varnothing \subseteq \{a, b, c\}$$

Since the empty set doesn't have any members, it never contains anything that is not an element of another set, so the definition of subset is always trivially satisfied. So whenever anybody asks you, "Is the empty set a subset of...?", you can answer "yes" without even hearing the rest of the sentence (though it's probably best to let them finish the question, just to be polite). On the other hand, if they ask you whether the empty set is an *element* of some other set, then you'll have to look among the elements of that other set in order to decide.

Proper subset. By this definition, every set is actually a subset of itself, even though normally we might first think of two sets of different sizes when we think of the subset relation. So the following

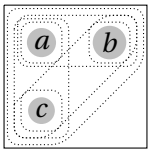


Figure 2.2: The powerset of $\{a, b, c\}$

statement is true:

$$\{a, b, c\} \subseteq \{a, b, c\}$$

To avoid confusion, it helps to distinguish between subsets and *proper* subsets. A is a PROPER SUBSET of B, written $A \subset B$, if and only if A is a subset of B and A is not equal to B. Here and throughout, we abbreviate 'if and only if' as *iff*:

$$A \subset B \text{ iff (i) for all } x\text{: if } x \in A \text{ then } x \in B \text{ and (ii) } A \neq B.$$

For example, $\{a, b, c\} \subseteq \{a, b, c\}$ but it is not the case that $\{a, b, c\} \subset \{a, b, c\}$.

Powerset. When we collect all of the subsets (proper or not) of a given set S into a set, that is called the POWERSET of S, written $\wp(S)$ or sometimes 2^S. The latter notation is motivated by the fact that if a set has n elements, then its powerset has 2^n elements. For example, if a set has 2 elements then its powerset has 4 elements:

$$\wp(\{a, b\}) = \{\{\}, \{a\}, \{b\}, \{a, b\}\}$$

Figure 2.2 depicts the powerset of $\{a, b, c\}$. It has $2^3 = 8$ elements, including the empty set $\varnothing$ and the set itself $\{a, b, c\}$.

Superset. The reverse of subset is superset. A is a SUPERSET of B, written $A \supseteq B$, if and only if every member of B is also in A.

$$A \supseteq B \text{ iff for all } x\text{: if } x \in B \text{ then } x \in A.$$

Proper superset. As you might expect, A is a PROPER SUPERSET of B, written $A \supset B$, if and only if A is a superset of B and A is not equal to B.

$$A \supset B \text{ iff (i) for all } x\text{: if } x \in B \text{ then } x \in A \text{ and (ii) } A \neq B.$$

Disjoint. If two sets have no elements in common, then they are called DISJOINT. Two sets with an empty intersection are disjoint. A and B are disjoint iff $A \cap B = \varnothing$.

2.2.3 Operations on sets

Relations among sets vs. operations on sets. Subsethood is a RELATION BETWEEN SETS, which can either hold or fail to hold. Other elements of the set theoretic vocabulary express OPERATIONS ON SETS, producing a new set from one or more sets. The principal operations on sets include intersection, union, and complement.

Intersection. The INTERSECTION of A and B, written $A \cap B$, is the set of all entities x that are both a member of A and a member of B.

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

Examples:

$$\{a, b, c\} \cap \{b, c, d\} = \{b, c\}$$

$$\{b\} \cap \{b, c, d\} = \{b\}$$

$$\{a\} \cap \{b, c, d\} = \varnothing$$

$$\{a, b\} \cap \{a, b\} = \{a, b\}$$

Union. Another useful operation on sets is union. The UNION of A and B, written $A \cup B$, is the set of all things that are either in A or in B (or both).

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

For example:

$$\{a, b\} \cup \{d, e\} = \{a, b, d, e\}$$
$$\{a, b\} \cup \{b, c\} = \{a, b, c\}$$
$$\{a, b\} \cup \varnothing = \{a, b\}$$

Exercise 2. Use D to denote the set of doctors and L to denote the set of lawyers. Which of the following best captures the meaning of *Anna is a doctor or a lawyer*?

(a) $a \in (D \cap L)$

(b) $a \in (D \cup L)$

(c) $a \subseteq (D \cap L)$

(d) $a \subseteq (D \cup L)$

What if *or* was replaced by *and*? Which of the above options best captures the meaning of *Anna is a doctor and a lawyer*?

Exercise 3. Use D to denote the set of doctors, L to denote the set of lawyers, and R to denote the property of being rich. Which of the following best captures the meaning of *Every doctor and every lawyer is rich*?

(a) $(D \cap L) \subseteq R$

(b) $(D \cup L) \subseteq R$

(c) $R \subseteq (D \cap L)$

(d) $R \subseteq (D \cup L)$

Set difference. We can also talk about SUBTRACTING one set from another. The DIFFERENCE of A and B, written $A - B$ or $A \setminus B$, is the set of all things that are in A but not in B.

$$A - B = \{x \mid x \in A \text{ and } x \notin B\}$$

For example, $\{a, b, c\} - \{b, d, f\} = \{a, c\}$. This is also known as the RELATIVE COMPLEMENT of A and B, or the result of subtracting B from A. $A - B$ can also be read, 'A minus B'. Sometimes people speak simply of the COMPLEMENT of a set A, without specifying what the complement is relative to. This is still implicitly a relative complement; it is relative to some assumed UNIVERSE or DOMAIN of entities. The complement of A can be written $\bar{A}$.[2]

Exercise 4. Fill in the rest of Table 2.1.

2.2.4 Subset vs. element

The exercises below are taken from Partee et al. 1990, *Mathematical Methods in Linguistics*. Cautionary note: There is an important difference between being an *element* of a set and being a *subset* of a set. The set $\{2,3\}$ is not an *element*, but rather a *subset* of the set $\{2,3,4\}$. On the other hand, $\{b\} \in \{a, \{b\}, c\}$.

Exercise 5. Given the following sets:

[2] The notation A' is also sometimes used for the complement.

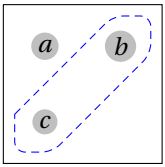
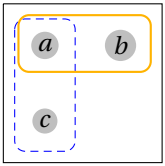
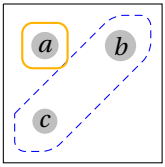
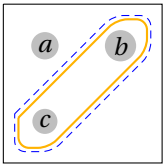
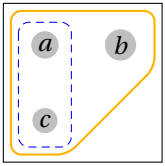
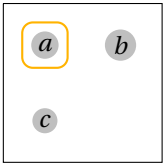
1		$S = \{b, c\}$	$S \cap P = \varnothing$	$\overline{S} = \{a\}$	$P \cap \overline{S} = \varnothing$
		$P = \varnothing$	$S \cup P = \{b, c\}$	$\overline{P} = \{a, b, c\}$	$S \cap \overline{P} = \{b, c\}$
2		$S = \{a, c\}$		$\overline{S} =$	$S \cap \overline{P} =$
		$P = \{a, b\}$		$\overline{P} =$	$P \cap \overline{S} =$
3		$S = \{b, c\}$		$\overline{S} =$	$S \cap \overline{P} =$
		$P = \{a\}$		$\overline{P} =$	$P \cap \overline{S} =$
4		$S = \{b, c\}$		$\overline{S} =$	$S \cap \overline{P} =$
		$P = \{b, c\}$		$\overline{P} =$	$P \cap \overline{S} =$
5		$S = \{a, c\}$		$\overline{S} =$	$S \cap \overline{P} =$
		$P = \{a, b, c\}$		$\overline{P} =$	$P \cap \overline{S} =$
6		$S = \varnothing$		$\overline{S} =$	$S \cap \overline{P} =$
		$P = \{a\}$		$\overline{P} =$	$P \cap \overline{S} =$

Table 2.1: Set-theoretic properties of various cases.

$A = \{a, b, c, 2, 3, 4\}$ $E = \{a, b, \{c\}\}$
$B = \{a, b\}$ $F = \varnothing$
$C = \{c, 2\}$ $G = \{\{a, b\}, \{c, 2\}\}$
$D = \{b, c\}$

classify each of the following statements as true or false.

(a) $c \in A$ (g) $D \subset A$ (m) $B \subseteq G$
(b) $c \in F$ (h) $A \subseteq C$ (n) $\{B\} \subseteq G$
(c) $c \in E$ (i) $D \subseteq E$ (o) $D \subseteq G$
(d) $\{c\} \in E$ (j) $F \subseteq A$ (p) $\{D\} \subseteq G$
(e) $\{c\} \in C$ (k) $E \subseteq F$ (q) $G \subseteq A$
(f) $B \subseteq A$ (l) $B \in G$ (r) $\{\{c\}\} \subseteq E$

Exercise 6. Given the sets $A, \ldots, G$ from above, repeated here:

$A = \{a, b, c, 2, 3, 4\}$ $E = \{a, b, \{c\}\}$
$B = \{a, b\}$ $F = \varnothing$
$C = \{c, 2\}$ $G = \{\{a, b\}, \{c, 2\}\}$
$D = \{b, c\}$

list the members of each of the following:

(a) $B \cup C$ (g) $A \cap E$ (m) $B - A$
(b) $A \cup B$ (h) $C \cap D$ (n) $C - D$
(c) $D \cup E$ (i) $B \cap F$ (o) $E - F$
(d) $B \cup G$ (j) $C \cap E$ (p) $F - A$
(e) $D \cup F$ (k) $B \cap G$ (q) $G - B$
(f) $A \cap B$ (l) $A - B$

2.3 Relations and functions

Sets give us tools for describing the PROPERTIES that individuals have, such as the property of being a pirate. Properties are attributes that an entity can have or lack. Another important type of information that can be encoded by expressions of natural languages is the RELATIONS that individuals stand in to each other. For example, to say that Anna is Ben's sister is to say that Anna stands in the 'sister' relation to Ben. To say that Anna loves Ben is to say that Anna stands in the 'love' relation to Ben. Relations can be modelled mathematically using pairs of elements that stand in a specified order to each other, i.e. ORDERED PAIRS.

2.3.1 Ordered pairs

As stated above (p. 53), sets are not ordered. For any a and b:

$$\{a, b\} = \{b, a\}$$

But the elements of an ORDERED PAIR are ordered. Using angle brackets, we write

$$\langle a, b\rangle$$

to designate the ordered pair in which a is the FIRST MEMBER and b is the SECOND MEMBER. Thus:

$$\langle a, b\rangle \neq \langle b, a\rangle$$

Like the elements of sets, the members of an ordered pair can be anything. Here is an ordered pair of numbers:

$$\langle 3, 4\rangle$$

A member of an ordered pair could also be a set, as in the ordered pair whose first member is the set $\{1,2,3\}$ and whose second member is the set $\{2,3,4\}$, written:

$$\langle \{1,2,3\}, \{2,3,4\}\rangle$$

Alternatively, one or both of the members could be ordered pairs, as in the following:

$$\langle 3, \langle 10, 12 \rangle \rangle$$

In this ordered pair, the first member is the number 3, and the second member is the ordered pair $\langle 10, 12 \rangle$. Note that $\langle 3, \{10, 12\} \rangle$ is not the same thing as $\langle 3, \langle 10, 12 \rangle \rangle$. The first is an ordered pair whose second member is the *set containing* 10 and 12; the second is an ordered pair whose second member is the *ordered pair* $\langle 10, 12 \rangle$.

Given two sets A and B, the set of ordered pairs $\langle x, y \rangle$ such that $x \in A$ and $y \in B$ is called the CARTESIAN PRODUCT of A and B, written $A \times B$. For example:

$$\{a, b, c\} \times \{1, 2, 3\}$$

$$= \{\langle a, 1 \rangle, \langle a, 2 \rangle, \langle a, 3 \rangle, \langle b, 1 \rangle, \langle b, 2 \rangle, \langle b, 3 \rangle, \langle c, 1 \rangle, \langle c, 2 \rangle, \langle c, 3 \rangle\}$$

Exercise 7. True or false?

(a) $\{3, 3\} = \{3\}$

(b) $\{3, 4\} = \{4, 3\}$

(c) $\langle 3, 4 \rangle = \langle 4, 3 \rangle$

(d) $\langle 3, 3 \rangle = \langle 3, 3 \rangle$

(e) $\{\langle 3, 3 \rangle\} = \langle 3, 3 \rangle$

(f) $\{\langle 3, 3 \rangle, \langle 3, 4 \rangle\} = \{\langle 3, 4 \rangle, \langle 3, 3 \rangle\}$

(g) $\langle 3, \{3, 4\} \rangle = \langle 3, \{4, 3\} \rangle$

(h) $\{3, \{3, 4\}\} = \{3, \{4, 3\}\}$

2.3.2 Relations

2.3.2.1 Types of relations

As mentioned above, the semantics of transitive verbs like *love*, *admire*, and *respect* is sometimes modelled using RELATIONS between two individuals. The 'love' relation corresponds to the set of ordered pairs of individuals such that the first member loves the second member. Suppose John loves Sandy. Then the pair ⟨John, Sandy⟩ is a member of this relation.

Certain nouns, including *neighbor*, *mother*, and *friend*, can be thought of as denoting relations between individuals. So can prepositions like *in* and *beside*. Relations can also hold between sets; for example, *subset* is a relation between two sets A and B which holds if and only if every element of A is an element of B. As mentioned before, this is arguably the relation expressed by the determiner *every*; if every A is a B, then A is a subset of B.

A preposition like *in* denotes a relation between *two* individuals; that is, it denotes a BINARY RELATION. The preposition *between*, by contrast, expresses a TERNARY RELATION, that is, a relation between three objects (a is between b and c). A ternary relation can be modelled as a set of ordered triples. For example, the ternary relation denoted by *between* contains the following triples:

$$\langle \text{Alabama}, \text{Mississippi}, \text{Georgia} \rangle$$

$$\langle \text{Togo}, \text{Ghana}, \text{Benin} \rangle$$

as Alabama is between Mississippi and Georgia and Togo is between Ghana and Benin. A QUATERNARY relation corresponds to a set of ordered 4-tuples. For example, it might be convenient for some purposes to consider a 'spatiotemporal location' relation that holds between an entity, a latitude, a longitude, and a time.

Given sets A and B, a RELATION FROM A TO B is a set of ordered pairs whose first member is an element of A and whose second member is an element of B; formally, it is a (proper or non-proper) subset of the Cartesian product $A \times B$. The set A is

called the DOMAIN of the relation, and the set B its CODOMAIN. Not all elements of the domain and codomain need be involved in the relation: those elements of B that actually occur as a second member of some pair form the RANGE (also called the IMAGE) of the relation. A relation from S to S is called a RELATION ON S.

The sets A and B can be, but need not be distinct. One can also be a subset of the other. A REFLEXIVE relation is one that relates everything to itself, that is, for any x, the pair $\langle x, x\rangle$ is in the relation. (Other pairs may be in the relation, too.) For example, the relation 'greater than or equal to' is reflexive, because every number is greater than or equal to itself.

A relation is SYMMETRIC if and only if: For any a and b, if $\langle a, b\rangle$ is in the relation, then $\langle b, a\rangle$ is also in the relation. For example, the 'next to' relation is symmetric; hence the following argument is valid:

(1) Paul is next to George.
$\therefore$ George is next to Paul.

The 'admires' relation is not, though.

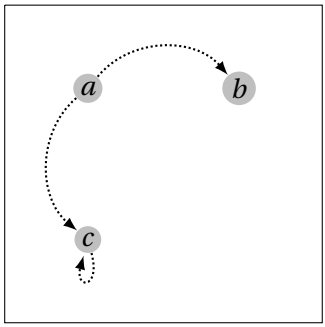
(2) Paul admires George.
$\not\therefore$ George admires Paul.

Here we see an example of how mathematical properties of the relations expressed by words and phrases in natural language can affect the inference patterns that they license.

A TRANSITIVE relation is one that licenses inferences like this:

(3) Paul is taller than George.
George is taller than Ringo.
$\therefore$ Paul is taller than Ringo.

In general, a relation is TRANSITIVE if and only if: For any x, y, and z, if $\langle x, y\rangle$ and $\langle y, z\rangle$ are in the relation, then $\langle x, z\rangle$ is also in the relation. (This notion TRANSITIVE should not be confused with



$R = \{\langle a, c\rangle, \langle c, c\rangle, \langle a, b\rangle\}$

Figure 2.3: A visualization of a relation using arrows

the notion of a transitive verb.) Another example of a transitive relation is 'before': If x is before y, and y is before z, then x is before z.

A relation can be visualized using arrows. Figure 2.3 shows an example of a relation $R = \{\langle a, c\rangle, \langle c, c\rangle, \langle a, b\rangle\}$ on the set $\{a, b, c\}$. We can think of this as the 'is a fan of' relation; a is a fan of b and c, c is a fan of only herself, and b is a fan of nobody. R is *not* symmetric, because, for example, $\langle a, b\rangle$ in the relation while $\langle b, a\rangle$ is not. R *is* transitive, because there are no elements x, y and z in the relation such that $\langle x, y\rangle$ and $\langle y, z\rangle$ are in the relation, but $\langle x, z\rangle$ is not. For instance, one way to instantiate x, y and z so that $\langle x, y\rangle$ and $\langle y, z\rangle$ are in the relation is: $x = a$, $y = c$, $z = c$. Here $\langle x, z\rangle$ is in the relation because $\langle a, c\rangle$ is in the relation. R is not reflexive because $\langle a, a\rangle$ and $\langle b, b\rangle$ are not in it (although $\langle c, c\rangle$ is, so the relation is not anti-reflexive).

A relation that is reflexive, symmetric, and transitive is called an EQUIVALENCE RELATION. For example, the relation 'has the same birthday as' is an equivalence relation. So is 'sibling'. An equivalence relation determines a PARTITION over a set, that is, a set of non-intersecting subsets that cover the whole set (so the union of the subsets is equal to the whole set). Each member of

the partition is called a CELL of the partition. So, for example, if we group people by birthday, we can form a partition over the set of people with a number of cells equal to the number of different birthdays. Within each cell, the elements will stand in the 'have the same birthday' equivalence relation to each other, and it is in that sense that the equivalence relation determines the partition. This notion comes up in the analysis of questions, although we will not touch on that in this book.

Exercise 8. One of the following arguments is valid, and the other is not.

(i) The singer is the drummer's brother.
∴ The drummer is the singer's brother.

(ii) The singer is the drummer's sibling.
∴ The drummer is the singer's sibling.

Which one is valid? Why is it valid while the other is not? Put your answer to the second question in the following form: "Because *sibling* expresses a _____ relation and _____ does not."

Exercise 9. One of the following arguments is valid, and the other is not.

(i) The singer is immediately to the left of the drummer.
The drummer is immediately to the left of the lead guitarist.
Therefore, the singer is immediately to the left of the lead guitarist.

(ii) The singer is to the left of the drummer.
The drummer is to the left of the lead guitarist.
Therefore, the singer is to the left of the lead guitarist.

Which one is valid? Why is it valid while the other is not? Put your answer in the following form: "Because _____ expresses a _____ relation and _____ does not."

Exercise 10. ABBA is composed of two couples: Björn and Agnetha, and Frida and Benny. The 'partner' relation over the members of ABBA can be expressed as the following set of pairs:

$\{\langle \text{Agnetha}, \text{Björn}\rangle, \langle \text{Björn}, \text{Agnetha}\rangle, \langle \text{Frida}, \text{Benny}\rangle, \langle \text{Benny}, \text{Frida}\rangle\}$

(a) Is the 'partner' relation symmetric? Explain why or why not.

(b) Is the 'partner' relation transitive? Explain why or why not.

2.3.3 Functions

We turn now to functions, a special type of relation. The word 'function' has many senses, but here we are using it in its mathematical sense. You can think of a mathematical function as something like a vending machine: It takes one or more INPUTS (e.g. a specification of which item you would like to buy, and a means of payment), and returns an OUTPUT (e.g. a particular bag of chocolate-covered raisins). The inputs to functions are also called ARGUMENTS (this is unrelated to the notion of an *argument* as constituted by a series of statements that we encountered in Chapter 1). The outputs of functions are also called VALUES.

An example of a function is a relation that maps a person to their height in feet and inches. For example, given Michelle Obama (the person herself) it returns $5'11''$ (five feet and 11 inches). Because functions are relations, a function is essentially a set of ordered pairs. The following ordered pairs are members of this 'height' function:

$$\langle \text{Michelle Obama}, 5'11'' \rangle$$

$$\langle \text{Angela Merkel}, 5'5'' \rangle$$

$$\langle \text{Jacinda Ardern}, 5'5'' \rangle$$

Every function is a relation (by definition), but not every relation is a function. A relation from A to B is a FUNCTION only if every element of A is mapped to one and *only* one member of B. In the example at hand, we have a relation from people to heights. Two different people may be mapped to the same height, but for every person, there is only one height that it maps to. For example, both Angela Merkel and Jacinda Ardern (former political leaders of Germany and New Zealand, respectively) are mapped to $5'5''$ by this 'height' function, but the only value that Angela Merkel is mapped to is $5'5''$. An example of a relation that is *not* a function is the 'sister' relation, because a single person may have multiple sisters. In Figure 2.4, the relations depicted are *not* functions. In Figure 2.5, the relations depicted *are* functions.

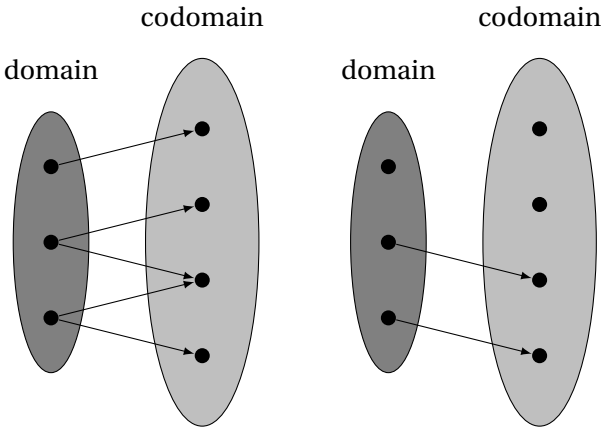


Figure 2.4: Two non-functions

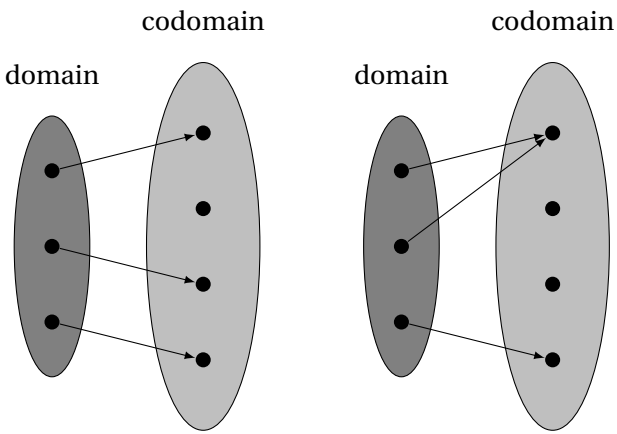


Figure 2.5: Two functions

Functions can be written either as a set of ordered pairs:

$$\{\langle \text{M. Obama}, 5'11''\rangle, \langle \text{Angela Merkel}, 5'5''\rangle, \langle \text{Jacinda Ardern}, 5'5''\rangle, ...\}$$

or using large brackets like this:

$$\left[\begin{array}{lcl} \text{Michelle Obama} & \rightarrow & 5'11'' \\ \text{Angela Merkel} & \rightarrow & 5'5'' \\ \text{Jacinda Ardern} & \rightarrow & 5'5'' \\ ... & & \end{array}\right]$$

The style with large brackets is easier to read (though not as easy to type), so we will often use that style.

We write $f(a)$ for 'the result of applying function f to argument a' or 'f of a' or 'f applied to a'. In this PARENTHESIS NOTATION, the argument is enclosed within parentheses. Note that there are no spaces surrounding the parentheses. If f is a function that contains the ordered pair $\langle a, b\rangle$, then:

$$f(a) = b$$

This means that given a as input, f gives b as output. More properly speaking, we say that a is given to f as an ARGUMENT, and that b is the VALUE of the function f when a is given as an argument.

Exercise 11. Some nouns in English express relations; these are called *relational nouns*. A special class of relational nouns expresses functions; these are sometimes called *functional nouns*. For example, *mother* (in the biological sense) is a functional noun (assuming that the relevant domain consists of people) because every person has a unique mother. On the other hand, *aunt* is not, because some people have multiple aunts or none at all. Which of the following might be called functional nouns? When answering this question, assume domains where the relations in question are defined. For example, when deciding whether *height* is a function, assume a domain that only contains objects that can have height to begin with.

(a) *height*

(b) *center*

(c) *edge*

(d) *part*

(e) *age*

(f) *citizenship*

Given a set A, a function that takes an entity and returns 1 (True) if that entity is a member of A and 0 (False) otherwise is called the CHARACTERISTIC FUNCTION of A. For example, the set of ABBA band members is $\{$Agnetha, Björn, Benny, Frida$\}$. With this set as the domain, the characteristic function of the set $\{$Agnetha, Frida$\}$ is a function that takes as input an ABBA member and re-

turns 1 (True) if the input is a member of this set and 0 if not:

$$\begin{bmatrix} \text{Agnetha} & \rightarrow & 1 \\ \text{Björn} & \rightarrow & 0 \\ \text{Benny} & \rightarrow & 0 \\ \text{Frida} & \rightarrow & 1 \end{bmatrix}$$

This function, applied to Agnetha, yields 1 (True). Applied to Björn, it yields 0 (False). (Conversely, if f is the characteristic function of S, then S is the CHARACTERISTIC SET of f.) A set and its characteristic function can be converted into each other without losing information. For this reason, semanticists often switch back and forth between sets and functions without being explicit about it.

The denotations (in a particular case) of common nouns like *tiger* and *picnic* and *student* are sometimes treated as sets – the set of tigers, the set of picnics, the set of students. But characteristic functions provide an equivalent way of capturing the same information. This fact will turn out to be convenient as we develop our semantic theory in later chapters.

Exercise 12. Recall that ABBA is composed of two couples—Björn and Agnetha, and Frida and Benny—and that the 'partner' relation over the members of ABBA can be expressed as the following set of pairs:

$\{\langle \text{Agnetha}, \text{Björn}\rangle, \langle \text{Björn}, \text{Agnetha}\rangle, \langle \text{Frida}, \text{Benny}\rangle, \langle \text{Benny}, \text{Frida}\rangle\}$

(a) The 'partner' relation (on the set of ABBA members) is a function. If we call this partner function f, then we can use the parentheses notation to designate the value of the function when applied to an argument. For example, we can write $f(\text{Agnetha})$ to designate the value of the 'partner of' function when applied to Agnetha. What is the value of $f(\text{Agnetha})$?

(b) True or false: $f(\text{Björn}) = \text{Frida}$

(c) True or false:
$f(f(\text{Björn})) = f(\text{Agnetha})$

Exercise 13. Recall that the characteristic function of a set is a function that maps every member of that set to 1, and every non-member (in some specified larger set) to 0. For example, the characteristic function of the set of female individuals in ABBA is:

$$\{\langle \text{Agnetha}, 1\rangle, \langle \text{Björn}, 0\rangle, \langle \text{Benny}, 0\rangle, \langle \text{Frida}, 1\rangle\}$$

(a) Give the characteristic function of the set of male individuals in ABBA.

(b) Call the function you defined in the previous question male. What is the value of male(Björn)?

(c) Suppose that the verb phrase *is male* denotes this function male. Suppose further that the name *Björn* denotes Björn Ulvaeus of ABBA. Suppose that the denotation of a sentence consisting of a noun phrase and a verb phrase is the result of applying the function denoted by the verb phrase to the denotation of the noun phrase. What, then, is the denotation of the sentence *Björn is male*? (Give the value of the function.)

(d) Under the same assumptions (plus the assumption that *Agnetha* denotes Agnetha Fältskog of ABBA), what is the denotation of the sentence *Agnetha is male*?

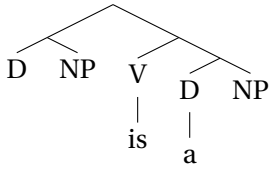
2.4 Quantification in English

2.4.1 A theory of quantification in English

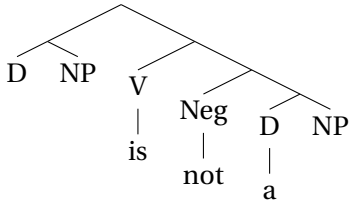
We are now ready to give a first semantic theory for a fragment of English. Since we haven't started developing our representation language yet (a variant on Alonzo Church's typed lambda calculus, which we finally build up to in Chapter 5), we will state the theory using only the meta-language (a version of English that is enriched by the preceding set-theoretic notions).

We will give semantics for a tiny little fragment of English, just comprising sentences of the following two forms:

(4) a.



b.



where NP can be lexicalized as either *sailor* or *pirate* and the initial D can be lexicalized as *every*, *some*, or *no*. This mini-grammar generates all four corners of the square of opposition, along with some additional sentences, such as:

(5) a. No pirate is not a sailor.
b. Every sailor is a sailor.
c. Some pirate is a sailor.

Exercise 14. Name two more sentences that the mini-grammar generates, besides A *Every sailor is a pirate*, I *Some sailor is a pirate*, E *No sailor is a pirate*, and O *Some sailor is not a pirate*, and the ones just given in (5).

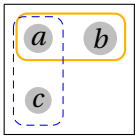
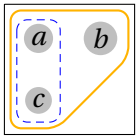
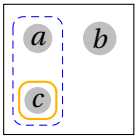
i.	ii.	iii.
		
$S = \{a, c\}$ $P = \{a, b\}$	$S = \{a, c\}$ $P = \{a, b, c\}$	$S = \{a, c\}$ $P = \{c\}$

Table 2.2: Three cases

Let us assume that *sailor* and *pirate* pick out (or DENOTE) sets. It varies from case to case who is a sailor, so the set that *sailor* picks out may vary from case to case. The set that *sailor* picks out in a given case can be called the EXTENSION of *sailor* in that case. A noun might have an empty extension in one case, and a non-empty extension in another. Let us use the labels S and P for the sets they pick out in a given case.

Let us assume that S and P are sets of INDIVIDUALS, drawn from a DOMAIN OF INDIVIDUALS. Let us assume further that there are just three individuals to pick from: a, b, and c. Several cases obeying these constraints are illustrated in Table 2.2.

The determiners *every*, *some* and *no* express relations that either hold or do not hold between the two sets. Here is a simple theory of the meanings of *every* and *some*:

(6) a. *Every sailor is a pirate* expresses that S is a subset of P:

$$S \subseteq P$$

More generally, *Every X is a Y* expresses that X is a subset of Y:

$$X \subseteq Y$$

b. *Some sailor is a pirate* expresses that the intersection

between S and P is non-empty:

$$S \cap P \neq \varnothing$$

More generally, *Some X is a Y* expresses that the intersection between X and Y is non-empty:

$$X \cap Y \neq \varnothing$$

(Here we are playing a bit fast and loose by allowing X and Y to be part of both the object language and the meta-language, for the sake of simplicity. We will be stricter about this later.)

Under this analysis, *Every sailor is a pirate* is PREDICTED TO BE TRUE in case (ii), since there S is indeed a subset of P. The same sentence is PREDICTED TO BE FALSE in cases (i) and (iii), where S contains elements that are not in P. Keep in mind that predictions are different from facts: The predictions of any given analysis may or may not match up with the empirical facts stemming from the intuitions of native speakers of the language. Different analyses may be better or worse at capturing the facts, and our job as scientists is to discover the facts and match our theories to them.

Exercise 15. Add a line to (6) for the determiner *no*, in a manner that captures its requirement for an empty intersection.

Now what does negation express in *Some sailor is **not** a pirate*? Here is an intuitive suggestion: Whatever set *pirate* denotes, *not a pirate* denotes its complement, containing all non-pirates.

(7) If *NP* denotes set X, then the denotation of *not a NP* is

$$\bar{X}$$

With the appropriate fixing up of our semantic rules to handle negated predicates, this will lead to an analysis under which *Some*

sailor is not a pirate expresses that the intersection between S and the complement of P is non-empty:

$$S \cap \bar{P} \neq \varnothing$$

Exercise 16. Match the corners of the square of opposition (A, I, E, and O) to a truth condition (i-ix) as expressed in set theory notation. Assume that S is the set of sailors and P is the set of pirates. Not every option will be used.

A Every sailor is a pirate.

I Some sailor is a pirate.

E No sailor is a pirate.

O Some sailor is not a pirate.

(i) $S \cap \bar{P} \neq \varnothing$ (iv) $S \cap P \neq \varnothing$ (vii) $S \subseteq P$

(ii) $S \cup P = \varnothing$ (v) $S \cup \bar{P} \neq \varnothing$ (viii) $S \in P$

(iii) $S \cap \bar{P} = \varnothing$ (vi) $S \subset P$ (ix) $S \cap P = \varnothing$

2.4.2 Generalized quantifier theory

What do all English determiners have in common?[3] The following schema reveals a striking answer:

(8) _____ A is/are B iff _____ A is/are A that B.

All of the determiners we have encountered so far satisfy this:

(9) a. Every dog barks iff every dog is a dog that barks.

[3]The treatment of generalized quantifier theory in this section draws on Partee (2008), which provides an excellent pedagogical overview.

b. Some dog barks iff some dog is a dog that barks.
c. No dog barks iff no dog is a dog that barks.

This says that the truth of a sentence 'Det A B' depends only on which members of the restrictor A are also in the nuclear scope B — not on anything in B that falls outside A. This property is called CONSERVATIVITY, and it is crucial to a striking empirical finding of GENERALIZED QUANTIFIER THEORY (GQT), the research program that investigates which relations among sets can be denoted by natural language determiners.[4]

Conservativity. Generalized quantifier theory views the meanings of quantificational determiners as relations among sets, and asks about the properties of these relations. For example, *every* denotes the relation that holds of two sets A and B iff $A \subseteq B$; *some* denotes the intersection relation, holding of A and B iff $A \cap B \neq \varnothing$.

One question that can be asked about a given relation among sets is whether it is CONSERVATIVE. Formally, a Det Q is CONSERVATIVE if

$$Q(A)(B) \Leftrightarrow Q(A)(A \cap B)$$

for all sets A and B, where the double arrow $\Leftrightarrow$ can be read as 'if and only if'. In other words, the verdict of Q depends only on which members of the restrictor are in the nuclear scope, not on what lies in the nuclear scope outside the restrictor. Keenan & Stavi (1986) proposed the CONSERVATIVITY UNIVERSAL: all natural language determiners, in any language, denote conservative Dets.

[4] The name comes from the observation that the denotation of a DP like *every cat* can be represented as a set of sets of individuals — specifically, the set of all supersets of the cat-set. Such an object is called a GENERALIZED QUANTIFIER because the standard first-order quantifiers $\forall$ and $\exists$ can be similarly represented: $\exists$ corresponds to the set of all non-empty subsets of D_e, and $\forall$ to the singleton $\{D_e\}$. Determiners then denote binary relations between sets. One payoff of this perspective is a uniform semantic type for all DPs: *John*, *every cat*, and *three dogs* all denote sets of sets — the set of John's properties, the set of all supersets of the cat-set, and the set of all sets containing at least three dogs, respectively.

The equivalences in (9) can each be verified using the formal definition. *Every* is conservative because it only cares whether all members of A fall in B: replacing B with $A \cap B$ asks whether all members of A fall in $A \cap B$, which is equivalent — any member of A that is in B is automatically in $A \cap B$. For *some*, both $Q(A)(B)$ and $Q(A)(A \cap B)$ ask whether A and B share a member; since $A \cap (A \cap B) = A \cap B$, the two conditions are identical.

Exercise 17. Using the entry $Q(A)(B) = 1$ iff $A \cap B = \varnothing$ for *no*, verify that *no* denotes a conservative Det by showing that $Q(A)(B) \Leftrightarrow Q(A)(A \cap B)$ for all sets A and B.

An apparent counterexample comes from relative measures such as *percent* and *mostly* (Ahn & Sauerland, 2015). Compare:

(10) a. The company hired 55% of the women.(conservative)
b. The company hired 55% WOMEN. (non-conservative)

Reading (10)a is conservative: it asks what fraction of the women were hired. Reading (10)b, however, can mean that women constitute 55% of those hired — a reading that depends on who among the company's hires is *not* a woman, not just on the intersection of women with hires. This appears to violate conservativity. Ahn & Sauerland 2015 argue, however, that the non-conservative reading arises not from a non-conservative determiner but from a distinct syntactic structure involving covert movement of the measure noun, together with focus semantics. These examples are therefore consistent with the Conservativity Universal after all.

Cardinal and proportional determiners. Milsark (1977) observed that determiners vary as to which can serve as the pivot (the DP immediately following *there is/are*) in EXISTENTIAL-THERE CONSTRUCTIONS:

(11) a. There are some cats in the garden.

b. There are three cats in the garden.
c. *There is every cat in the garden.
d. *There are most cats in the garden.

He called those that can occur 'weak' and those that cannot 'strong'. It turns out that this empirical distinction maps very neatly onto a theoretical one. Here are some 'weak' determiners, together with their denotations:

- *some*: $Q(A)(B) = 1$ iff $A \cap B \neq \varnothing$
- *no*: $Q(A)(B) = 1$ iff $A \cap B = \varnothing$
- *three*:[5] $Q(A)(B) = 1$ iff $|A \cap B| \geq 3$

These are CARDINAL in the GQT sense: their verdicts depend only on the cardinality of $A \cap B$. To see this, notice that each entry references only properties of the intersection — whether it is nonempty, empty, or of a specific size — and nothing about A or B individually. A consequence is SYMMETRY: since the verdict depends only on $|A \cap B| = |B \cap A|$, swapping the restrictor and the nuclear scope cannot change it:

(12) a. Three students are musicians iff three musicians are students.

[5] A note on terminology: CARDINAL NUMERALS (e.g. *three*, *seven*) are so called in morphosyntax to distinguish them from ORDINAL NUMERALS (*third*, *seventh*). This is a purely distributional classification and is independent of the GQT sense of *cardinal* defined below. The situation is further complicated by *many*, which has both a cardinal reading (*many students came* — a large absolute number) and a proportional reading (*many of the students came* — a large proportion); and *most*, which has both 'relative' and 'proportional' readings, a range that can be captured under the assumption that *most* is the superlative of *many* rather than a simple binary relation between sets (Hackl, 2009). The entry given here treats *three* as 'at least three'; the stronger 'exactly three' understanding is standardly analyzed as a scalar implicature (Horn, 1972; Geurts, 2011). Other analyses are on the market, however, including ones on which number words are scope-taking degree quantifiers, so that their lower-bounded, upper-bounded and two-sided readings all arise semantically rather than through implicature (Kennedy, 2015).

b. Some students are musicians iff some musicians are students.

On the other hand, the following determiners are not cardinal:

- *every*: $Q(A)(B) = 1$ iff $A \subseteq B$
- *most*: $Q(A)(B) = 1$ iff $|A \cap B| > |A - B|$

To illustrate, swapping the restrictor and nuclear scope of *every* does not preserve truth:

(13) a. Every student is a musician $\not\leftrightarrow$ every musician is a student.

The non-cardinal determiners in the preceding list share a different property: they are PROPORTIONAL. A Det Q is proportional iff $Q(A)(B)$ depends on the cardinality of $A \cap B$ relative to the cardinality of the restrictor A. For example, *every* is proportional because its verdict holds iff $A \subseteq B$ — that is, iff the entire restrictor is contained in the nuclear scope — which depends not just on $|A \cap B|$ but on how $|A \cap B|$ compares to $|A|$. Every cardinal determiner is symmetric, since its verdict depends only on $|A \cap B| = |B \cap A|$; though the converse does not hold in general.

Exercise 18. The determiner *three* is a cardinal numeral. Is it also a cardinal Det in the GQT sense? Use the symmetry diagnostic to decide: does $Q(A)(B) = Q(B)(A)$ for all sets A and B, where Q is the denotation of *three*?

2.5 Summary

We opened this chapter with the square of opposition, noting that Aristotle's four categorical propositions — *every S is P*, *some S is P*, *no S is P*, *some S is not P* — express relations among categories,

and that these relations can be visualized with Euler diagrams. The central insight is that the meanings of natural language quantifiers are fundamentally set-theoretic: understanding what *every* and *some* mean requires understanding what it means for one set to be a subset of another, or for two sets to have a non-empty intersection.

To make this precise, we developed the necessary formal toolkit. We introduced sets and the membership relation, operations on sets (union, intersection, complement, set difference), and the relations that can hold between sets (subset, superset, equality, disjointness). We also drew a careful distinction between the subset relation, which holds between sets, and the membership relation, which holds between an element and a set — a distinction that is easy to blur but important to keep straight.

We then extended our attention from sets to *relations* — sets of ordered pairs — and *functions*, which are relations satisfying the additional constraint that each input is paired with exactly one output. Relations can have structural properties such as reflexivity, symmetry, and transitivity, which will matter both for the analysis of relational nouns (explored in the problem sets below) and for later work on formal semantics. Functions in particular will be central to everything that follows: the semantic value of a transitive verb, a determiner, or any complex expression will be modelled as a function.

Finally, we applied this toolkit directly to natural language quantification through Generalized Quantifier Theory. Treating determiners as binary relations between sets allowed us to state two cross-linguistic generalizations — the Conservativity Universal and the distinction between cardinal and proportional determiners — with mathematical precision. The empirical weak/strong distinction observed by Milsark (1977) in existential-there constructions falls out as a reflex of the cardinal/proportional distinction.

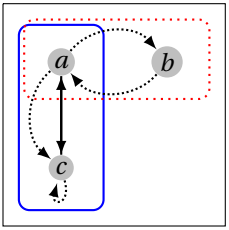
The chapters immediately ahead introduce formal logical languages for reasoning about propositions. In Chapter 3, we de-

velop propositional logic, and in Chapter 4, predicate logic, which will allow us to represent the internal structure of sentences. The set-theoretic concepts introduced here serve as the metalanguage for both: propositions will be interpreted as sets of circumstances, and the entailment relation between propositions corresponds to the subset relation between such sets.

Problem Set 2A: Reciprocal plurals

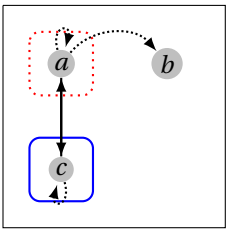
Case 1

$M = \{a, c\}$
$F = \{a, b\}$
$R_{\text{sib}} = \{\langle a, c\rangle, \langle c, a\rangle\}$
$R_{\text{fan}} = \{\langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle, \langle c, c\rangle\}$



Case 2

$M = \{c\}$
$F = \{a\}$
$R_{\text{sib}} = \{\langle a, c\rangle, \langle c, a\rangle\}$
$R_{\text{fan}} = \{\langle a, a\rangle, \langle a, b\rangle, \langle c, c\rangle\}$



Case 3

$M = \varnothing$
$F = \{a, b, c\}$
$R_{\text{sib}} = \{\langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle, \langle c, a\rangle, \langle b, c\rangle, \langle c, b\rangle\}$
$R_{\text{fan}} = \varnothing$

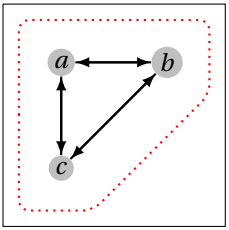


Table 2.3: Three cases involving relations. M - solid blue; F - dotted red; sibling - solid double arrow; 'is a fan of' - dotted arrow.

Table 2.3 illustrates three cases involving two relations and two properties. The individuals in M, the set of male individuals, are encircled by a solid blue line. The individuals in F, the set of female individuals, are encircled by a dotted red line. The sibling relation is indicated with a solid double arrow, and the 'is fan of' relation is indicated with a dotted arrow.

Exercise 2A-1. Suppose that 'brother' denotes the set of pairs $\langle x, y\rangle$ such that x is male and x and y stand in the sibling relation: $\{\langle x, y\rangle \mid x \in M \text{ and } \langle x, y\rangle \in R_{\mathsf{sib}}\}$

Refer to Table 2.3. In Case 1, which of the following pairs are in this relation, if any? (Recall that a relation is a set of ordered pairs.)

i. $\langle a, a\rangle$ iv. $\langle b, a\rangle$ vii. $\langle c, a\rangle$

ii. $\langle a, b\rangle$ v. $\langle b, b\rangle$ viii. $\langle c, b\rangle$

iii. $\langle a, c\rangle$ vi. $\langle b, c\rangle$ ix. $\langle c, c\rangle$

Exercise 2A-2. This question is the same as the previous question, but it asks about Case 2 instead of Case 1.

Exercise 2A-3. In which of the Cases 1-3 does it hold that:

$$\{x \mid x \in M \text{ and } \langle x, c\rangle \in R_{\mathsf{fan}}\} \neq \varnothing$$

(This may hold in multiple cases, or none.)

Exercise 2A-4. In which of the cases in Table 2.3 does it hold that:

$$\{x \mid \langle x, x\rangle \in R_{\mathsf{fan}}\} = \{a, c\}$$

(This may be true in multiple cases, or none.)

Exercise 2A-5. Please rate the following statements as true or false relative to the three cases in Table 2.3.

1. In every case, the sibling relation is symmetric.
2. In every case, the fan relation is symmetric.
3. The sibling relation is not symmetric in any of the cases.
4. The fan relation is not symmetric in any of the cases.

Exercise 2A-6. Here are 10 nouns that denote binary relations.

(i) sibling

(ii) sister

(iii) Facebook friend

(iv) dogwalker

(v) mother

(vi) Instagram follower

(vii) spouse

(viii) wife

(ix) secret admirer

(x) acquaintance

Which of these relations are symmetric? Recall: A relation R is SYMMETRIC if for every pair $\langle x, y\rangle$ that is in R there is a pair $\langle y, x\rangle$ that is also in R.

Assume that the field of the relation (the set of entities that might appear as the first or second member of any of the pairs in the relation) in each case is the set of human beings that exist in the actual world at the moment you are reading this. (One can't be presumed to know all the relations that all billions of us stand in to each other, but in answering this question, one can rely on their knowledge of the meanings of the words to make assumptions about that.)

Exercise 2A-7. Which of the relations listed in the previous exercise are asymmetric, if any? Recall: A relation R is ASYMMETRIC iff: for all x, y: If $\langle x, y\rangle$ is in R, then $\langle y, x\rangle$ is *not* in R.

There is a construction in English involving a plural relational noun like *sibling*, illustrated in (14).

(14) A and B are siblings.

On the most prominent reading of this example, it implies that A and B are *each other's* siblings. Let us refer to this as a RECIPROCAL READING. (i) also has a non-reciprocal reading, which could be brought out by the continuation, "...so they both deserve to be celebrated on Siblings' Day". (This type of reading could be derived via a meaning-changing operation that converts *sibling* from a relational noun into a sortal noun by existentially quantifying over one of the participants in the relation.)

To bring out the reciprocal reading in context, we can position (14) as the answer to the question, "What relation do A and B

stand in to each other?"

(15) Harry: What relation to A and B stand in to each other?
Jess: A and B are *siblings*.

In this context, the reciprocal reading is the only one available. Staroverov (2007) refers to usages of plural nouns like *sibling* with this sort of reciprocal interpretation as RECIPROCAL PLURALS; we will follow this convention. (Schwarz 2006 uses the term 'covert reciprocals', which might be used for a broader class of such items.)

Not all nouns can be used to form reciprocal plurals. Here is a minimal pair:

(16) Harry: What relation do A and B stand in to each other?
Jess: They are *Facebook friends*.

(17) Harry: What relation do A and B stand in to each other?
Jess: #They are *Instagram followers*.

The hash-mark on the response with Instagram followers is an indication that the response is inappropriate in the context (a.k.a. *pragmatically infelicitous*).

One difference between *Facebook friend* and *Instagram follower* is that the former is a symmetric relation but the latter is not. Could that be responsible for the difference in acceptability between (16) and (17)? Let us call this the SYMMETRY HYPOTHESIS for reciprocal plurals: In order to form a reciprocal plural, a noun must denote a *symmetric relation*. Specifically:

(18) **Symmetry hypothesis for reciprocal plurals**
For any given singular noun N: If N denotes a symmetric relation, then 'X and Y are Ns' (where 'N*s*' is the plural of N) can have a reciprocal reading. Otherwise, it can't.

For example, *Facebook friend* denotes a symmetric relation and *Instagram follower* does not: *A is B's Facebook friend* entails *B is A's Facebook friend*, while *A is B's Instagram follower* does not entail *B*

is A's Instagram follower. The symmetry hypothesis for reciprocal plurals thus predicts that *A and B are Facebook friends* has a reciprocal reading while *A and B are Instagram followers* does not. This prediction is correct.

Exercise 2A-8. Let us further evaluate the predictions of the symmetry hypothesis. Which of the nouns below is/are predicted to form reciprocal plurals (a reciprocal reading of 'X and Y are Ns') under the symmetry hypothesis? (Select all that apply.)

(i) sibling

(ii) sister

(iii) dogwalker

(iv) mother

(v) spouse

(vi) wife

(vii) secret admirer

(viii) acquaintance

When doing this exercise, it is important to keep in mind the distinction between the plural nouns that occur in the sentences and their corresponding singular forms. The symmetry hypothesis says that a noun has a reciprocal *plural* use if and only if the *singular* form denotes a symmetric relation.

Exercise 2A-9. The symmetry hypothesis predicts that some of the sentences in Exercise 2A-8 have reciprocal readings and some do not. These predictions are *accurately borne out by the data* to the extent that sentences that really do have reciprocal readings are correctly predicted to have them, and sentences that really lack them are correctly predicted to lack them. Incorrect predictions can be either false positives (when a sentence is *predicted* to *have* a reciprocal reading when it *actually lacks* one), or false negatives (when a sentence is predicted to *lack* a reciprocal reading when it actually *has* one). For some of the sentences in the previous exercise, the symmetry hypothesis might make incorrect predictions. Your task in this exercise is to evaluate where it succeeds and where it fails.

(a) First, establish the empirical facts about which nouns allow reciprocal readings (or do your best to do so). To test whether a reciprocal reading is available, you can put the 'A and B are Ns' sentence in a context that brings out the reciprocal reading, as a response to 'What relation do A and B stand in to each other?' Try this test with each of the nouns in Exercise 2A-8. It might help to find a native speaker of English who hasn't been thinking about this for too long to give you their intuition about whether the constructed dialogue makes sense.

(b) Next, for each of the nouns in Exercise 2A-8, decide whether the symmetry hypothesis makes a correct or incorrect prediction. Summarize your findings by sorting the nouns into four cases:

- true positives: good, and correctly predicted to be good
- false positives: bad, but predicted to be good
- true negatives: bad, and correctly predicted to be bad
- false negatives: good, but predicted to be bad

Another hypothesis about reciprocal plurals is based on the idea that *brother* doesn't just entail 'male', it presupposes 'male' (and likewise for *sister* and 'female'). Suppose for the sake of argument that for all kinship terms that have a gender component, such as *sister*, *aunt*, *husband*, and *wife*, the gender component is presupposed. One way of capturing that formally is to say that *sister*, for example, denotes a relation R whose domain only consists of female individuals. In other words, it is a relation *from* the set of female individuals to the full set of individuals.

Schwarz (2006) suggests that "Strawson-symmetry" is the key property that allows a noun to be used as a reciprocal plural.[6]

(19) A relation is **Strawson-symmetric** iff: If $\langle x, y\rangle$ is in R, and y is in the domain of R, then $\langle y, x\rangle$ is in R.

For instance, let R_{bro} stand for the relation expressed by *brother*. This relation is Strawson-symmetric iff: If $\langle x, y\rangle$ is in R_{bro}, and y is in the domain of R_{bro}, then $\langle y, x\rangle$ is in R_{bro}. Under the assumption that the gender component is presupposed, the domain of the relation expressed by *brother* contains only male individuals. So in order to evaluate whether *brother* is Strawson-symmetric, it suffices to evaluate whether the following holds generally, for any A and B: If A is B's brother, *and B is male*, then B is A's brother. Indeed, this generalization is true. So the relation expressed by *brother* is Strawson-symmetric, under the assumption that the gender component is presupposed.

Suppose that what matters for the reciprocal plural construction is this property of Strawson symmetry.

(20) **Strawson-symmetry hypothesis for reciprocal plurals**
For any given singular noun N: If N denotes a Strawson-symmetric relation, then 'X and Y are Ns' (where 'Ns' is

[6] This name is an allusion to the term 'Strawson entailment' (see e.g. von Fintel 1999), which in a nutshell is entailment modulo presupposition, just as this is symmetry modulo presupposition.

the plural of *N*) can have a reciprocal reading. Otherwise, it can't.

As mentioned above, under the assumption that *brother* presupposes 'male', *brother* denotes a Strawson-symmetric relation. The Strawson-symmetry hypothesis for reciprocal plurals thus correctly predicts that *A and B are brothers* should have a reciprocal reading, under that assumption.

Exercise 2A-10. Which of the nouns in Exercise 2A-8 is predicted to form reciprocal plurals under the Strawson-symmetry hypothesis?

Problem Set 2B: Monotonicity and NPI Licensing

Certain words of English—including *ever*, *any*, and *anyone*—are called NEGATIVE POLARITY ITEMS (NPIs) because they are most naturally found in negative sentences. But negation is not the only environment that licenses them. This section develops the notion of MONOTONICITY and shows how it provides a principled account of NPI distribution in terms of the generalized quantifier semantics developed above.

Part 1: NPI licensing

The following contrast illustrates the basic phenomenon:

(21) a. Chrysler dealers don't *ever* sell *any* cars *anymore*.
b. *Chrysler dealers *ever* sell *any* cars *anymore*.

NPI licensing is not simply a matter of negation being present. The determiner *no* also licenses NPIs, but *some* and *every* behave differently depending on position. The relevant positions are the **NP position** (inside the determiner's restrictor) and the **VP position** (in its nuclear scope). The following paradigm, adapted from Ladusaw (1980), illustrates the contrast. In each example, (a) tests the **NP position** (inside the restrictor, marked with brackets) and (b) tests the **VP position** (in the nuclear scope, marked with brackets).

(22) a. No [student who had *ever* read *anything* about phrenology] attended the lecture.
b. No student [had *ever* read *anything* about phrenology].

(23) a. *Some [student who had *ever* read *anything* about phrenology] attended the lecture.
b. *Some student [had *ever* read *anything* about phrenology].

(24) a. Every [student who had *ever* read *anything* about phrenology] attended the lecture.
b. *Every student [had *ever* read *anything* about phrenology].

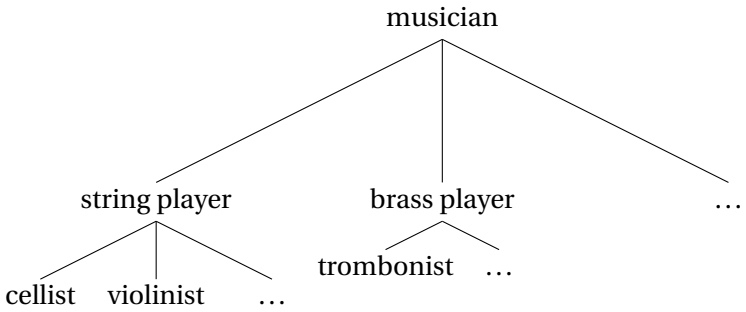
Exercise 2B-1. Based on the data in (22)–(24), fill in the following table, marking each cell with ✓ (NPIs licensed) or * (NPIs not licensed).

	NP position	VP position
no		
some		
every		

Is NPI licensing an all-or-nothing property of a determiner, or does it depend on position? What does this tell us about what the right generalization should look like?

Part 2: Upward and downward entailment

The relevant notion is MONOTONICITY: the direction in which a given position licenses entailments. To develop the idea, consider the following taxonomic hierarchy, with more specific categories below and more general ones above:



An environment *[… A …]* is UPWARD-ENTAILING (UE) in position A if replacing A with any more general term B (where $A \subseteq B$) preserves truth. It is DOWNWARD-ENTAILING (DE) in position A if replacing A with any more specific term B (where $B \subseteq A$) preserves truth.

The NP position of *some* is upward-entailing: the inference from a more specific restrictor to a more general one is valid.

(25) Some cellist attended.
∴ Some musician attended. (specific → general: valid ✓)

The VP position of *some* is likewise upward-entailing:

(26) Some musician snored loudly.
∴ Some musician snored. (specific → general: valid ✓)

By contrast, the NP position of *no* is downward-entailing: the inference goes from the more general term to the more specific one.

(27) No musician attended.
∴ No cellist attended. (general → specific: valid ✓)

And so is the VP position of *no*:

(28) No musician snored.
∴ No musician snored loudly. (general → specific: valid ✓)

Exercise 2B-2. For each environment below, determine whether it is UE or DE in the indicated position. Construct an inference of the form shown above to justify your answer. Use the pair {cellist, musician} to test the NP position (cellist ⊆ musician) and {snored loudly, snored} to test the VP position (snored loudly ⊆ snored).

(a) *every*: NP position. Evaluate: *Every musician attended*
∴ Every cellist attended.

(b) *every*: VP position. Evaluate: *Every musician snored*
∴ Every musician snored loudly. Then evaluate the reverse direction.

(c) Predicate negation (*not*, as in *Alex didn't snore*): VP position. Evaluate: *Alex didn't snore*
∴ Alex didn't snore loudly.

Part 3: The Fauconnier-Ladusaw generalization

We can now state the generalization connecting monotonicity to NPI licensing (Fauconnier, 1975; Ladusaw, 1980):

> **Fauconnier-Ladusaw generalization:** An expression α licenses NPIs in a position p if and only if p is a downward-entailing environment created by α.

Exercise 2B-3. (a) Confirm that the Fauconnier-Ladusaw generalization correctly predicts the NPI licensing pattern you tabulated in Exercise 2B-1. For each determiner and each position, state whether the environment is DE or UE, and whether

NPIs are therefore predicted to be licensed. Does the prediction match the data?

(b) The adverb *rarely* creates a downward-entailing VP environment. Predict whether NPIs should be licensed in sentences of the form *Casey rarely* __, then construct two examples to test your prediction.

(c) The *if*-clause of a conditional is a downward-entailing environment; the *then*-clause is not. Using an NPI of your choice, construct examples that confirm this contrast.

Part 4: Formal verification

Exercise 2B-4. The Fauconnier-Ladusaw generalization can be verified formally using the set-theoretic entries from Section 2.4.2. Recall:

- *no*: $Q(A)(B) = 1$ iff $A \cap B = \varnothing$
- *every*: $Q(A)(B) = 1$ iff $A \subseteq B$

Two proof templates will be useful. To prove $P \subseteq Q$: take an arbitrary element $d \in P$ and show that $d \in Q$. To prove $P \cap Q = \varnothing$: suppose for contradiction that some $d \in P \cap Q$ exists, then derive a contradiction from the assumptions.

(a) **NP position of *no* is DE.** Suppose $A \cap B = \varnothing$, and suppose $C \subseteq A$. Prove that $C \cap B = \varnothing$ follows.

(b) **VP position of *every* is UE.** Suppose $A \subseteq B$, and suppose $B \subseteq C$. Prove that $A \subseteq C$ follows.

(c) **VP position of *every* is not DE.** Construct a counterexample showing that even if $A \subseteq B$ and $C \subseteq B$, it need not follow that $A \subseteq C$. (Hint: let A be the set of musicians, B the set of individuals who snored, and C the set of individuals who snored loudly.)

3 Propositional logic

3.1 Introduction

In Chapter 1, we suggested that one of the things a good theory of meaning should capture is when one sentence entails another. For example, a good theory of meaning should correctly predict that the following are valid arguments:

(1) If it rained last night, then the lawn is wet.
It rained last night.
∴ The lawn is wet.

(2) Every man is mortal.
Socrates is a man.
∴ Socrates is mortal.

(3) Aristotle taught Alexander the Great.
Alexander the Great was a king.
∴ Aristotle taught a king.

We will start working in simplified settings. We will use artificial FORMAL LANGUAGES that are inspired by natural language but are also carefully designed to avoid much of its complexity and ambiguity. We will then design a semantics that systematically associates sentences in these formal languages with different truth values corresponding to different interpretations of the placeholders in these sentences. This will allow us to develop a notion of entailment. A formal language that is equipped with a

notion of entailment, and a way to determine when that notion applies, is called a LOGIC. The first formal language we will consider is PROPOSITIONAL LOGIC (also called SENTENTIAL LOGIC). In propositional logic, placeholders stand for entire clauses or sentences, so we can express arguments like (1) but not like (2) or (3). In Chapter 4, we will then introduce PREDICATE LOGIC (also called QUANTIFIER LOGIC), in which these latter two arguments can be expressed.

We will use logic to interpret natural language in a two-step manner (INDIRECT INTERPRETATION). First, we translate natural language into a logic, and then we interpret that logic, as explained in Chapter 1. By associating sentences of natural language with sentences of logic, and letting the entailment relation on sentences of natural language be inherited from the corresponding logic, we can provide a theory of entailment in natural language.

3.2 Propositional logic

Recall from the introduction that one of the main driving questions in the study of logic is: Under what conditions is an argument *valid*? For instance, the following argument (repeated here from (1)) is clearly valid:

(4)	If it rained last night, then the lawn is wet.	(Premise 1)
	It rained last night.	(Premise 2)
	∴ The lawn is wet.	(Conclusion)

This argument has two premises and a conclusion. The conclusion is a necessary consequence of the premises: As long as the premises are both true, the conclusion must be true too. One might disagree with the premises, but it does not matter whether they are actually true. As long as they are both granted, the conclusion holds. Hence, the argument is valid.

The conclusion in (4) is also a logical consequence of the premises. The argument is an instance of the argument form known as MODUS

PONENS:

(5) If p, then q.
p.
$\therefore q$.

Exercise 1. Using (4) as inspiration, give another argument using Modus Ponens.

Now consider this superficially similar argument:

(6) If it rained last night, then the lawn is wet. (Premise 1)
The lawn is wet. (Premise 2)
$\not\therefore$ It rained last night. (Conclusion)

One might be tempted to think that this argument is valid, but it is not. The premises might be true while the conclusion is false. It may well be true that the lawn gets wet whenever it rains, and that the lawn is wet. But if something other than rain can cause the lawn to become wet, perhaps a sprinkler, then the conclusion might still be false. Because the conclusion is not entailed by the premises taken together, the argument is not valid. This argument form, which is called AFFIRMING THE CONSEQUENT, is not correct—it is a FALLACY (an argument form that is not valid):

(7) If p, then q.
q.
$\not\therefore p$.

(Terminological note: in a conditional sentence of the form 'if p then q', p is called the ANTECEDENT and q is called the CONSEQUENT. The name of this fallacious argument form derives from the fact that the consequent is affirmed as a premise, and is then used to derive the antecedent of the conditional sentence.)

Exercise 2. Give another fallacious argument using Affirming the Consequent.

Propositional logic aims to capture the difference between correct argument forms and fallacies, focusing on ones that involve placeholders standing for clauses and sentences. By translating sentences into propositional logic, and building our notion of entailment on that of propositional logic, we can get a step closer towards developing a theory of entailment for natural language.

Exercise 3. For both of the following argument forms, say whether it is valid or a fallacy.

1. **Modus Tollens**
   If it rained last night, then the lawn is wet.
   The lawn is not wet.
   Therefore, it did not rain last night.

2. **Denying the antecedent**
   If it rained last night, then the lawn is wet.
   It didn't rain last night.
   Therefore, the lawn is not wet.

3.2.1 Formulas and propositional letters

Let us begin our introduction to propositional logic with the notion of a PROPOSITIONAL LETTER (also called *propositional variable* or *sentential letter*). A propositional letter is a symbol that represents roughly the kind of thing that is expressed by a simple declarative clause or sentence that does not contain any of the words *and, or, not, if, then.* For example, the propositional letter p is a placeholder for clauses like "Boston is the capital of Nebraska",

or "red is a primary color", or any other sentence of this nature that is either true or false. In this chapter, we will adopt the following inventory of propositional letters:

Syntactic Rule: Propositional letters
p, q, and r are propositional letters.

(A summary of definitions like this will be compiled in Section 3.3.2.)

Other choices would also be possible within the realm of what is called 'propositional logic'. In principle, any set of symbols can be used as propositional letters. When more letters are needed, it is customary to use primes as in p, p', p'', etc. Different choices of letters will give rise to different PROPOSITIONAL LANGUAGES. We will refer to the specific propositional language we are building up as $\mathrm{L}_{\mathrm{Prop}}$.

Just like natural languages, propositional languages and other logics consist of grammatical sentences. In the context of logic, it is common to use the term WELL-FORMED rather than "grammatical". The counterpart in logic of a grammatical sentence is called a FORMULA or WFF (for *well-formed formula*). Our mapping from natural languages to representation languages will map natural language sentences to logical formulas (or *formulae*, as the plural of *formula* is sometimes written) with the same denotations as the sentences.

Now, what is the denotation of a formula? Frege suggested that the denotation of a natural language sentence is a truth value: True (T) or False (F).[1] In order to know if a formula is true or false, we need to know which of its propositional letters we should consider true and which ones we should consider false. An INTERPRETATION FUNCTION, sometimes just called INTERPRETATION, for a

[1] In Chapter 8, we will countenance a third truth value (Neither), but here we stick to classical logic, which has just two.

given propositional language is a function that maps each propositional letter of that language to a truth value. What are these interpretations? There are different ways to think about them, and we come back to this below. But what they have in common is that an interpretation provides enough information to determine truth values for all the formulas of propositional logic. Here is an example of an interpretation function for $\mathrm{L}_{\mathrm{Prop}}$.

$$\begin{bmatrix} p & \to & \mathsf{T} \\ q & \to & \mathsf{T} \\ r & \to & \mathsf{F} \end{bmatrix}$$

This says that p is true, and that q is true, while r is false.

We will speak of formulas being true or false "under an interpretation" (that is, given an interpretation function) or "with respect to an interpretation". In propositional logic, an interpretation relative to which a given formula is true coincides with what is called a MODEL FOR that formula. (Later, when we get to predicate logic, the notions of 'model' and 'interpretation' will come apart; as we will see, a model will then be taken to specify both an interpretation and certain additional information that is not yet relevant now.)

Any propositional letter, taken by itself, is a formula. But just as natural language expressions may be built up from smaller expressions, formulas in propositional logic may be also built up from smaller formulas. To define and interpret formulas of arbitrary size, we will lay down SYNTACTIC RULES (also called *rules of formation*) and SEMANTIC RULES. Syntactic rules specify how to build formulas, and semantic rules specify how to interpret them, that is, how to map them to T or F. The semantic rules will be compositional, in the sense that they assign denotations to larger formulas in ways that depend only on the denotations of the smaller formulas (rather than on their shape or length, for example).

As we assign truth values to complex formulas in terms of smaller ones, we will introduce a DENOTATION FUNCTION, which provides

a denotation to every formula of the language, by extending a given interpretation function which just assigns denotations to the propositional letters. The denotation function is written using double square brackets (a.k.a. 'semantic brackets'), and carries a superscript in order to specify the interpretation function it is based on:

Notational convention
For any well-formed formula ϕ of propositional logic, let $[\![\phi]\!]^I$ stand for the denotation of ϕ with respect to interpretation function I.

Here, the Greek letter ϕ ("phi") is a META-VARIABLE, a symbol which stands for a formula of propositional logic. Typical meta-variables for propositional logic include ϕ (sometimes written φ) and ψ ("psi").[2] Meta-variables are not themselves part of L_{Prop}; they are part of the META-LANGUAGE that we use to talk about L_{Prop} and other logics. (Recall from Chapter 1 that we said that our meta-language would be English with some mathematical jargon mixed in; meta-variables are among that mathematical jargon.)

Recall the example interpretation function given above:

$$\begin{bmatrix} p & \rightarrow & \mathsf{T} \\ q & \rightarrow & \mathsf{T} \\ r & \rightarrow & \mathsf{F} \end{bmatrix}$$

Call this I_1. Then, for example, $[\![p]\!]^{I_1}$ ('the denotation of p under I_1') is T. Thus, interpretation functions and denotation functions both map formulas to their truth values. The difference is that interpretation functions only apply to propositional letters, while denotation functions apply to all formulas of our propositional language. The interpretation function I will typically differ

[2] The Greek letter phi, written ϕ, looks very similar to the empty set symbol $\varnothing$, but this is just an accident; they are completely unrelated symbols.

from one propositional language to the other, while the denotation function $[\![\cdot]\!]^I$ extends I in a completely predictable way. Different propositional languages could have different propositional letters or could use the same letter for different purposes, in which case their interpretations I will have to differ. But once I is fixed, $[\![\cdot]\!]^I$ is fixed too: the denotations of larger formulas are derived entirely from those of the propositional letters they contain. (The difference between I and $[\![\cdot]\!]^I$ in logic is like the difference between lexicon and compositional semantics in English. If the meaning of a given English word were to change, the lexicon of English would change to reflect that fact, but the compositional semantics of English would stay the same.)

The following semantic rule specifies that in the case of propositional letters, I and $[\![\cdot]\!]^I$ coincide, for any interpretation function I:

Semantic Rule: Propositional letters
If ϕ is any propositional letter and I is any function from proposition letters to truth values, then

$$[\![\phi]\!]^I = I(\phi)$$

So, for example, $[\![p]\!]^I = I(p)$. If $I(p) = \mathsf{T}$, then $[\![p]\!]^I = \mathsf{T}$ as well.

Exercise 4. Let I_1 be defined as above. What is the value of $I_1(r)$? What is the value of $[\![r]\!]^{I_1}$?

3.2.2 Boolean connectives

Formulas in propositional logic can be combined and assembled into larger formulas by using the so-called LOGICAL CONNECTIVES, or just CONNECTIVES. These connectives correspond roughly to

the English expressions *and*, *or*, *not*, *if . . . then*, and *if and only if* (which, as in earlier chapters, we often abbreviate *iff*). The meanings of these expressions are intimately connected with each other. To illustrate: Suppose you ask your friend, "Are you free *today or tomorrow*?" and she says *no*. That means that she's *not* free today, *and* she's not free tomorrow. In general, the following argument form is valid:

(8) not [p or q]
$\therefore$ [not p] and [not q]

as is its converse,

(9) [not p] and [not q]
$\therefore$ not [p or q]

Because the argument is valid in both directions,

[not p] and [not q]

and

not [p or q]

are EQUIVALENT, a relationship we can express using 'if and only if' (in a sense to be distinguished below from the truth-functional connective of the same name):

(10) [[not p] and [not q]] if and only if [not [p or q]]

Now, suppose you ask your friend, "Are you free today *and* tomorrow?" and she says *no*. That is not quite as strong; it means that *either* she's not free today *or* she's not free tomorrow (or both). Thus the following argument form is valid:

(11) not [p and q]
$\therefore$ [not p] or [not q]

Its converse is valid as well:

(12) [not p] or [not q]
$\therefore$ not [p and q]

Again, we have an equivalence:

(13) [[not p] or [not q]] if and only if [not [p and q]]

The equivalences in (10) and (13) are called DE MORGAN'S LAWS. We elaborate on them a few pages down.

By specifying a syntax and an interpretation for connectives corresponding to *and*, *or*, and *not*, we can capture the logical relationships between these words.

The term CONNECTIVE is used in logic for symbols that connect formulas, or attach to them, to form new formulas. A propositional letter standing alone is called an ATOMIC FORMULA, while formulas that are formed with the help of connectives are called COMPLEX FORMULAS. Two examples of connectives in propositional logic are the symbol $\wedge$ (sometimes written &), pronounced 'and', and the symbol $\vee$ (sometimes written $|$), pronounced 'or'. Symbols such as conjunction and disjunction are called BINARY CONNECTIVES, because they join two formulas together. The negation symbol $\neg$ (sometimes written $\sim$) is called a UNARY CONNECTIVE, because it applies to a single formula to produce a new one. Connectives, particularly unary ones, are also called OPERATORS.

Consider the sentence *Susan does not volunteer on Monday*. This can be represented in propositional logic as follows. Let the propositional letter p represent the sentence *Susan volunteers on Monday*. We then represent *Susan does not volunteer on Monday* as follows:

$$\neg p$$

This is a formula and can be read 'it is not the case that p', or simply, 'not p'. The $\neg$ symbol represents 'it is not the case that'. In general:

Syntactic rule: Negation
If ϕ is a formula, then $\neg\phi$ is also a formula. (This is called the NEGATION of ϕ.)

Now, this $\neg$ symbol is interpreted in such a way that $\neg p$ is true whenever p is false, and vice versa. There are two possibilities to consider: p is true; p is false. The interpretation of $\neg$ can be represented using a TRUTH TABLE, as follows. A truth table is a way of representing interpretation functions and showing how the denotation function extends them to complex formulas. Each row in a truth table corresponds to a different interpretation function.

p	$\neg p$
T	F
F	T

This says: If p is true, then $\neg p$ is false; and if p is false, then $\neg p$ is true.

We will express the information contained in this truth table in a different format as our official semantic rule:

Semantic Rule: Negation
If ϕ is a formula, then $[\![\neg\phi]\!]^I = \mathsf{T}$ if $[\![\phi]\!]^I = \mathsf{F}$, and F otherwise.

The expression "and F otherwise" is a common shorthand that we will frequently use to indicate whatever is the relevant other possibility; here, for example, it stands for "and $[\![\neg\phi]\!]^I = \mathsf{F}$ if $[\![\phi]\!]^I = \mathsf{T}$".

Let us now consider the binary connectives, corresponding to *and* and *or*. An expression of the form 'X and Y' is called a CONJUNCTION; an expression of the form 'X or Y' is called a DISJUNCTION. In English, conjunctions can join two noun phrases, as in

Susan volunteers on Monday and Wednesday, where *Monday* and *Wednesday* are two noun phrases joined by *and*. But in this example, what is actually expressed can also be expressed as the conjunction of two sentences, which we can represent using the letters p and q. Let the propositional letter p represent the sentence *Susan volunteers on Monday* as above, and let the propositional letter q represent the sentence *Susan volunteers on Wednesday*. We can then represent *Susan volunteers on Monday and Wednesday* in propositional logic as follows:

$$[p \wedge q]$$

This is a formula and can be read 'p and q'. It is a CONJUNCTION in which p and q are the two CONJUNCTS. In general:

Syntactic rule: Conjunction
If ϕ and ψ are formulas, then $[\phi \wedge \psi]$ is also a formula.

This truth table for $\wedge$ is as follows:

p	q	$[p \wedge q]$
T	T	T
T	F	F
F	T	F
F	F	F

The semantic rule expresses the same information as the truth table in more compact form:

Semantic Rule: Conjunction
If ϕ and ψ are formulas, then $[\![[\phi \wedge \psi]]\!]^I = \mathsf{T}$ if $[\![\phi]\!]^I = \mathsf{T}$ and $[\![\psi]\!]^I = \mathsf{T}$, and F otherwise.

The DISJUNCTION of ϕ and ψ is written $[\phi \vee \psi]$. In such a formula, ϕ and ψ are called DISJUNCTS. For example:

$$[p \vee q]$$

can be read 'p or q'. In general:

Syntactic rule: Disjunction
If ϕ is a formula and ψ is a formula, then $[\phi \vee \psi]$ is also a formula.

The interpretation of $\vee$ can be represented as follows.

p	q	$[p \vee q]$
T	T	T
T	F	T
F	T	T
F	F	F

Semantic Rule: Disjunction
If ϕ and ψ are formulas, then $[\![[\phi \vee \psi]]\!]^I = \mathsf{T}$ if $[\![\phi]\!]^I = \mathsf{T}$ or $[\![\psi]\!]^I = \mathsf{T}$ (or both), and F otherwise.

This interprets $\vee$ as INCLUSIVE DISJUNCTION, because the statement is considered true even in the case where *both* of the disjuncts are true. This might surprise you. Suppose you heard this sentence:

(14) Susan volunteers on Monday or Wednesday.

Would you conclude that Susan volunteers on Monday or Wednesday, *but not both*? If so, then you are getting a so-called EXCLUSIVE interpretation, where the possibility that she volunteers on *both*

days is *excluded*. An INCLUSIVE interpretation is one on which the sentence is still true if she volunteers on both days.

EXCLUSIVE DISJUNCTION specifies that only one of the disjuncts is true. While it is not generally considered part of propositional logic, it would not be difficult to define an exclusive disjunction connective, sometimes written XOR (for *eXclusive OR*).

Exercise 5. Specify appropriate syntactic and semantic rules and an appropriate truth table for the exclusive disjunction connective XOR.

One might imagine that natural language *or* is ambiguous between inclusive and exclusive disjunction. But there is reason to believe that inclusive reading is what *or* really denotes, and that the exclusive reading arises via a conversational implicature. One argument for this comes from the fact that negation reliably brings out the inclusive disjunction (e.g. Horn, 1985; Schwarz et al., 2008). If I say *Kim did not invite Pat or Sandy*, it follows that Kim did not invite Pat and also did not invite Sandy. As for unembedded disjunctions, experiments have consistently shown that most of the time they are considered true, not false, when both disjuncts are true (e.g. Paris, 1973).

So far, we have discussed the semantics of $\wedge$, $\vee$, and $\neg$. In each case, the truth value of a complex expression that is produced by combining one of these connectives with the appropriate number of formulas depends solely on the truth values of their constituent formulas. Connectives with this property are called TRUTH-FUNCTIONAL. In Chapters 11 and 12, we will encounter connectives that are not truth-functional.

Truth tables can be used to compute the truth values for arbitrarily complex formulas using these connectives. For instance, let us consider when the formula $\neg[p \wedge q]$ is true. To find out, we first find out when $[p \wedge q]$ is true, and then apply negation to that.

p	q	$[p \wedge q]$	$\neg[p \wedge q]$
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

(The brackets [ ] are crucial here, as they show that we are applying negation to the conjunction of p and q. As we will see later, the syntax rules for propositional logic will ensure that a formula like $[\neg p \wedge q]$ would be interpreted as the conjunction of $\neg p$ and q.) The final column in the truth table above, for $\neg[p \wedge q]$, is the result of 'flipping' the truth values in the preceding column, for $[p \wedge q]$. This is what the truth table for negation tells us to do.

Recall that a sentence A entails a sentence B whenever in every case where A is true, B is true as well. Similarly, when A and B have the same truth values in every case, we say they are EQUIVALENT. We will write $A \models B$ to say that A entails B, and $A \equiv B$ to say that A and B are equivalent; the relationship we expressed with 'if and only if' in (10) is one we can now write as $\equiv$. Both of these symbols belong to our meta-language: unlike the connectives, they are not part of propositional logic itself, and they cannot be used to build formulas. A formula combines expressions of the object language, whereas $A \models B$ and $A \equiv B$ are statements we make *about* formulas. When our sentences are formulas of propositional logic, and our cases are interpretations, entailment and equivalence can be easily checked with truth tables, where every row corresponds to an interpretation. To do so, construct a truth table with columns for both formulas, and observe how the two columns relate. For example, to prove that p is equivalent to $\neg\neg p$, we can use the following truth table, where the columns for the two formulas in question are highlighted:

p	$\neg p$	$\neg\neg p$
T	F	T
F	T	F

Since this formula only has one propositional letter, we only need to consider two cases, each corresponding to a row of the truth table. The case where it's true corresponds to the first row, and the case where it's false corresponds to the second row. Observe that in the case where p is true, $\neg\neg p$ is also true, and in the case where p is false, $\neg\neg p$ is also false.

De Morgan's laws involve two propositional letters, so there are four cases to consider, as each propositional letter might be either true or false in a given interpretation. For instance, to prove that $\neg[p \wedge q]$ is equivalent to $[\neg p \vee \neg q]$, let us use the following truth table, where the columns for $\neg[p \wedge q]$ and $[\neg p \vee \neg q]$ are highlighted. (The non-highlighted columns are there as intermediate steps that will allow you to compute the highlighted columns, which are the main ones of interest.) As you can see, the pattern of T and F values in the two columns is the same.

p	q	$[p \wedge q]$	$\neg[p \wedge q]$	$\neg p$	$\neg q$	$[\neg p \vee \neg q]$
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

The two formulas are thus true under all the same interpretations, and false under all the same interpretations, and this shows that they are logically equivalent.

Exercise 6. Show that $\neg[p \vee q]$ is equivalent to $[\neg p \wedge \neg q]$, using a truth table. This is one of De Morgan's Laws. Here is a start:

p	q	
T	T	
T	F	
F	T	
F	F	

What should we observe about your truth table? In other words, what shows that the two formulas are equivalent?

Entailment can also be proven using truth tables. Recall the definition of entailment: *A entails B* if and only if *in any case where A is true, B is true too.* Truth tables list out various alternative possible scenarios, and each row of the truth table corresponds to a different imaginable circumstance. Example:

$$[p \wedge q]$$

entails

$$p$$

because there is no row where $[p \wedge q]$ is true but p is not.

p	q	$[p \wedge q]$
T	T	T
T	F	F
F	T	F
F	F	F

Exercise 7. Does p entail $[p \wedge q]$? Explain why or why not, using the truth table.

Exercise 8. Decide whether or not:

$$\neg[p \vee q]$$

entails

$$\neg[p \wedge q]$$

Start by filling in this truth table:

p	q	$[p \vee q]$	$\neg[p \vee q]$	$[p \wedge q]$	$\neg[p \wedge q]$
T	T				
T	F				
F	T				
F	F				

Based on the truth table you constructed, does $\neg[p \vee q]$ entail $\neg[p \wedge q]$? Explain.

Let us take a moment to reflect on the exact type of entailment that we have captured using these formal tools. There are two ways of viewing entailment, the first in terms of logical consequence and the second in terms of necessary consequence. The logical consequence view says that an argument is valid just in case there is no way to interpret its placeholders that results in an argument with a true premise and false conclusion. And the necessary consequence view says a valid argument is one for which there is no possible circumstance under which the premises are

true but the conclusion false. Which of these have we implemented here?

The answer depends on what interpretations are. Can we see interpretations as corresponding to specific possible worlds, or 'ways things could have been'? If so, we can think of the entailment relation of our logic as necessary consequence. Can we see interpretations as corresponding to specific ways to fill in the placeholders in an argument form? If so, we can think of our entailment relation as logical consequence. Because a model assigns truth values to different propositional letters independently of one another, the two perspectives are equivalent only as long as one assumes that the intrinsic meanings of propositional letters are independent of one another. In propositional logic, this assumption is always made. By contrast, if we were to take the letters p and q in a given language to stand for propositions that aren't both true in any possible world, this assumption would be violated. Suppose for example that we took p and q to stand for pairs of propositions such as "today is Tuesday" and "tomorrow is Tuesday"; or "this is red" and "this is colorless"; or "there is water in my cup" and "there is no H_2O in my cup"; or "John has grandchildren" and "John is childless". In all these cases, it seems that an interpretation function that maps both p and q to $\top$ does not correspond to any possible world. Now, the way we have set things up, the truth tables always list out all combinations of truth values for proposition letters like p and q. Each row corresponds to an 'interpretation' of the proposition letters, and to check entailment, we consider all of these interpretations, even if they are intuitively 'impossible' in some sense. Hence the notion of consequence that we have implemented here is logical consequence, rather than necessary consequence. In Chapter 12, we will bring necessary consequence back into the picture.

Truth tables can also help to shed light on SCOPAL AMBIGUITY. The following sentence is scopally ambiguous:

(15) Geordi didn't consult both Troi and Worf.

It can mean either of the following:

(16) a. Geordi consulted neither Troi nor Worf.
b. It is not the case that Geordi consulted both Troi and Worf (although he might have consulted one or the other).

The two readings can be modelled based on the relative scope of negation and conjunction. Assume that the propositional letter p stands for the English sentence *Geordi consulted Troi*, and the propositional letter r stands for the sentence *Geordi consulted Worf*.[3] The two readings can then be represented as:

(17) a. $\neg[p \wedge r]$
b. $[\neg p \wedge \neg r]$

These two formulas are not equivalent. However, the second formula is equivalent to $\neg[p \vee r]$, which explains why 'Geordi didn't consult Troi and Worf' can mean the same thing as 'Geordi didn't consult Troi or Worf'!

Exercise 9.

(a) Which of the formulae in (17) captures the reading in (16a)?

(b) Which corresponds to the reading in (16b)?

You might have noticed that we always place square brackets around conjunctions and disjunctions. In general, the outer

[3] We can do this without causing too much confusion here because *Geordi consulted Troi* and *Geordi consulted Worf* are logically independent of each other; they could both be true, they could both be false, or one or the other of them could be the only true one. If we were dealing with two sentences that were not independent in this way (e.g., if one entailed the other or they were mutually contradictory), then this type of 'translation' would lead us to consider impossible combinations of truth values for the sentences.

square brackets that go with binary connectives are always there according to the official rules of the syntax that we have defined here, though not necessarily in other presentations of propositional logic. We will sometimes drop them when they are not necessary for disambiguation. Sometimes, operator precedence rules are assumed. For example, in the absence of brackets, negation is taken to TAKE SCOPE UNDER (i.e. bind more strongly than) the binary connectives. (The SCOPE of a connective in a formula is the part of the formula that stands in for the metavariable(s) in its syntactic rule.) This means that a formula like $\neg p \wedge r$ is the conjunction of $\neg p$ with r, and is not equivalent to $\neg[p \wedge r]$. Likewise, the material conditional and the biconditional, which we are about to encounter, are sometimes taken to TAKE SCOPE OVER all other connectives. Brackets can be left in place to either override or reinforce these conventions. Conjunctions and disjunctions bind equally strongly, and one must take care to leave brackets in place. For example, $[[p \wedge q] \vee r]$ is not equivalent to $[p \wedge [q \vee r]]$. Here the brackets disambiguate: one should never write something like $[p \wedge q \vee r]$.

Exercise 10. An inclusive disjunction operator yields a formula that is true when both of the disjuncts are true. An exclusive disjunction operator yields a formula that is false unless exactly one of the disjuncts is true, so an exclusive disjunction formula is false when both of the disjuncts are true.

Does English *or* denote inclusive or exclusive disjunction? One way to tell is by looking at what happens under negation, as in *I don't have a doctor's appointment today or tomorrow*. How does this sort of example help to adjudicate between the two hypotheses about the semantics of English *or*? Include the relevant observation about the example and explain how it bears on the issue.

3.2.3 Conditionals and biconditionals

Recall that we want our logic to validate Modus Ponens ('If p then q; p; therefore q') as an argument form, but not Affirming the Consequent ('If p then q; q; therefore p'). In other words, we want to account for the fact that one is valid but not the other. There is a way of defining the semantics of CONDITIONAL statements (statements of the form 'if A then B') using truth tables that captures these facts. This method involves the so-called MATERIAL CONDITIONAL, a connective written as $\rightarrow$ (sometimes also $\supset$).

The material conditional is the truth-functional connective that comes closest to conditional statements (ones of the form 'if p then q') in natural language. Consider the following conditional sentence:

(18) If it's sunny, then it's warm.

(As a reminder, in a conditional sentence of the form 'if p then q', p is called the ANTECEDENT and q is called the CONSEQUENT. Here the antecedent is *it's sunny* and the consequent is *it's warm*.) There are four types of situations, in principle:

1. It's sunny and it's warm.
2. It's sunny and it's not warm.
3. It's not sunny and it's warm.
4. It's not sunny and it's not warm.

Let us consider which of these situations would falsify (18). Certainly the first situation does not. And if it's not sunny, then whether it's warm is irrelevant, because the claim only pertains to situations where it's sunny. So the third and the fourth situations would not falsify it. Since classical propositional logic has only two truth values, and we cannot plausibly assign F in these cases, we assign T instead. The only kind of situation that could falsify the claim is

the second one, where the antecedent is true and the consequent is false.

In general, a formula of the form $[p \to q]$ is false only when p is true and q is false, and true otherwise. The truth table for this connective looks like this:

p	q	$[p \to q]$
T	T	T
T	F	F
F	T	T
F	F	T

While it seems intuitively clear that a conditional is false when the antecedent is true and the consequent is false, it admittedly seems less intuitively clear that a conditional is *true* when the antecedent is false. For example, the moon is not made of green cheese. Does that mean that *If the moon is made of green cheese, then I had yogurt for breakfast this morning* is true? Intuitively not.

One might think that an indicative conditional is true only if the corresponding argument is valid. As we have seen there, an argument is not made valid merely by virtue of having a true conclusion; its validity depends on whether the conclusion is true in all cases where the premise is true. So one might reasonably argue that English indicative conditionals too cannot be judged as true or false based on a single case. In order to capture the truth conditions of indicative conditionals, we would need to talk about multiple circumstances or interpretations and not just a single one.[4] But the connectives of propositional logic are TRUTH-FUNCTIONAL: their truth value depends only on the truth values of their constituents. Among truth-functional connectives, the material conditional as we have defined it comes closest to doing the

[4] See Bennett (2003) and von Fintel (2011) for good introductions to the topic.

job. With it, we can account for the fact that Modus Ponens is valid and Denying the Antecedent is invalid.

Exercise 11. Fun fact: $[p \to q]$ is equivalent to $[\neg p \vee q]$. Show this by filling in the following truth table.

p	q	$[p \to q]$	$\neg p$	$[\neg p \vee q]$
T	T			
T	F			
F	T			
F	F			

What should we observe about this truth table? In other words, what shows that the two formulas are equivalent?

Exercise 12. Which of the following argument forms are valid in propositional logic (if any)?

1. $[p \to q]$
   p
   $\therefore q$
2. $[p \to q]$
   q
   $\therefore p$
3. $[p \to q]$
   $\neg q$
   $\therefore \neg p$

4. $[p \to q]$
   $\neg p$
   $\therefore \neg q$

As we have seen at the outset of this chapter, not all true conditionals have true converses. It may be true that *if it rained last night, the lawn is wet* and yet false that *if the lawn is wet, it rained last night*. But some conditionals do have the property that their converse holds:

(19) a. If yesterday was Sunday, then today is Monday.
b. If today is Monday, then yesterday was Sunday.

The logician's idiom *if and only if* can be used to succinctly express this kind of state of affairs:

(20) Today is Monday if and only if yesterday was Sunday.

The "if" part of this statement corresponds to (19a), which states that yesterday's being Sunday is a *sufficient condition* for today's being Monday. The "only if" part corresponds to (19b), or equivalently, to *Today is Monday only if yesterday was Sunday*, which states that yesterday's being Sunday is a *necessary condition* for today's being Monday. (By contrast, in gardens with sprinklers, its being rainy yesterday is typically a sufficient but not a necessary condition for the lawn's being wet today.) This "if and only if" formulation, sometimes abbreviated as "iff", is also used as a way of providing a definition of a concept with necessary and sufficient conditions.

This brings us to the last propositional logic connective we will introduce here: the BICONDITIONAL, written $\leftrightarrow$, and sometimes pronounced 'if and only if' (although as with the material conditional, it is just the closest thing to that idea we can express as a

truth-functional connective). $[p \leftrightarrow q]$ is true whenever p and q have the same truth value — either both true or both false. Its truth table looks like this:

p	q	$[p \leftrightarrow q]$
T	T	T
T	F	F
F	T	F
F	F	T

This truth table differs from that for $[p \rightarrow q]$ only in the third row. When p is false and q is true, $[p \rightarrow q]$ is true but $[p \leftrightarrow q]$ is false.

Notice, though, that the English sentence in (20) says more than the material biconditional does. In asserting *Today is Monday if and only if yesterday was Sunday*, a speaker claims a connection that holds in every case, not merely that the two sentences happen to agree in truth value on the day of utterance. The stronger relation is the meta-language equivalence $\equiv$ introduced earlier: a formula $[p \leftrightarrow q]$ can be true by coincidence, whereas $p \equiv q$ requires the two to agree under every interpretation. As we will see below, the two notions are related: $p \equiv q$ holds precisely when $[p \leftrightarrow q]$ is a tautology.

3.2.4 Equivalence, contradiction and tautology

As mentioned above, if two formulas are true under exactly the same interpretations, then they are EQUIVALENT. For example, p and $\neg\neg p$ are equivalent; whenever one is true, the other is true, and whenever one is false, the other is false, too:

p	$\neg p$	$\neg\neg p$
T	F	T
F	T	F

Exercise 13. Using truth tables, check whether the following pairs of formulas are equivalent.

(a) $[p \vee q]$; $\neg[\neg p \wedge \neg q]$

(b) $[p \to q]$; $[\neg p \vee q]$

(c) $\neg[p \wedge q]$; $[\neg p \vee \neg q]$

(d) $[p \vee \neg q]$; $\neg[p \wedge \neg q]$

(e) $[p \to q]$; $[\neg q \to \neg p]$

(f) $[p \to p]$; $[p \vee \neg p]$
(The truth table for this one should only contain two rows, since it doesn't mention q.)

Two formulas are CONTRADICTORY iff for every assignment of values to their variables, their truth values are different. For example p and $\neg p$ are contradictory.

p	$\neg p$
T	F
F	T

Another contradictory pair is $[p \to q]$ and $[p \wedge \neg q]$.

p	q	$[p \to q]$	$\neg q$	$[p \wedge \neg q]$
T	T	T	F	F
T	F	F	T	T
F	T	T	F	F
F	F	T	T	F

A TAUTOLOGY (also called *valid formula*) is a formula that is true under every assignment. The opposite, an expression that is false under every assignment, is called a CONTRADICTION; such a formula is also called *inconsistent* or *unsatisfiable*. Formulas that are neither valid nor inconsistent are called CONTINGENT, and formulas that are either valid or contingent are called SATISFIABLE. You can tell which of these categories a formula falls under by looking at the pattern of Ts and Fs in the column underneath it in a truth table: If they are all true, the formula is satisfiable and valid; if some are true and others are false, it is satisfiable and contingent; if they are all false, it is inconsistent. Here is a tautology: $[p \vee \neg p]$ (e.g. *It is raining or it is not raining*):

p	$\neg p$	$[p \vee \neg p]$
T	F	T
F	T	T

When two expressions are equivalent, the formula obtained by joining them with a biconditional is a tautology, and conversely. For example, $[p \leftrightarrow \neg\neg p]$ is a tautology:

p	$\neg p$	$\neg\neg p$	$[p \leftrightarrow \neg\neg p]$
T	F	T	T
F	T	F	T

Using the meta-language symbols introduced earlier, the connection can be stated in general terms: $A \equiv B$ if and only if $[A \leftrightarrow B]$ is a tautology, and $A \vDash B$ if and only if $[A \rightarrow B]$ is a tautology. Note the division of labour here: the biconditional and the conditional are connectives, used to build formulas of propositional logic, while $\equiv$ and $\vDash$ are used in the meta-language to say how such formulas behave across all interpretations.

Exercise 14. Which of the following are tautologies?

(a) $[p \vee q]$

(b) $[[p \rightarrow q] \vee [q \rightarrow p]]$

(c) $[[p \rightarrow q] \leftrightarrow [\neg q \vee \neg p]]$

(d) $[[[p \vee q] \rightarrow r] \leftrightarrow [[p \rightarrow q] \vee [p \rightarrow r]]]$

Support your answer with truth tables.

Exercise 15. Match each given formula of propositional logic (A.-D.) to a formula (i.-vii.) that is equivalent to it according to the rules of propositional logic. Use truth tables on a piece of scratch paper to check whether you are right. Not every formula (i.-vii.) will be used.

A. $\neg[p \vee q]$
B. $\neg[p \wedge q]$
C. $[p \rightarrow q]$
D. $[\neg p \rightarrow \neg q]$

i. $[p \vee q]$
ii. $\neg[p \rightarrow q]$
iii. $[q \rightarrow p]$
iv. $[\neg p \wedge \neg q]$
v. $\neg\neg[p \wedge q]$
vi. $[\neg p \vee \neg q]$
vii. $[\neg p \vee q]$

3.3 Summary: Propositional logic

To summarize what we have covered so far, we are defining here a simple propositional logic language called $\mathrm{L}_{\mathrm{Prop}}$. All languages of propositional logic are like this language up to the choice of propositional letters. We begin by listing all of the syntactic rules, to define what counts as a well-formed expression of the language, and then give the rules for semantic interpretation.

It is worth emphasizing that a logic is a *language* (or a class of languages), and comes with both syntax and semantics. The syntax specifies the well-formed formulas of the language. The semantics specifies the semantic value of every well-formed formula, given an interpretation.

3.3.1 Syntax of $\mathrm{L}_{\mathrm{Prop}}$

1. **Atomic formulas**

   - **Propositional letters:** p, q, r

2. **Complex formulas**

   - **Negation (Unary connective):** If ϕ is a formula, then $\neg\phi$ ('not ϕ') is a formula.
   - **Binary connectives**: If ϕ and ψ are formulas, then so are:
     - $[\phi \wedge \psi]$ 'ϕ and ψ'
     - $[\phi \vee \psi]$ 'ϕ or ψ'
     - $[\phi \rightarrow \psi]$ 'if ϕ then ψ'
     - $[\phi \leftrightarrow \psi]$ 'ϕ if and only if ψ'

The outer square brackets with binary connectives are always there according to the official rules of the syntax, but we sometimes drop them when they are not necessary for disambiguation.

3.3.2 Semantics of $\mathrm{L}_{\mathrm{Prop}}$

Let $[\![\phi]\!]^I$ stand for the denotation of a given expression ϕ with respect to an interpretation function I.

1. **Propositional letters**

   - If ϕ is any propositional letter, then

     $$[\![\phi]\!]^I = I(\phi).$$

2. **Complex formulas**

   - **Unary connective:** If ϕ is a formula, then $[\![\neg\phi]\!]^I = \mathsf{T}$ if $[\![\phi]\!]^I = \mathsf{F}$, and F otherwise.

3. **Binary connectives:** If ϕ and ψ are formulas, then:

   - $[\![[\phi \wedge \psi]]\!]^I = \mathsf{T}$ if $[\![\phi]\!]^I = \mathsf{T}$ and $[\![\psi]\!]^I = \mathsf{T}$, and F otherwise.
   - $[\![[\phi \vee \psi]]\!]^I = \mathsf{T}$ if $[\![\phi]\!]^I = \mathsf{T}$ or $[\![\psi]\!]^I = \mathsf{T}$ (or both), and F otherwise.
   - (Semantic rules for $\rightarrow$ and $\leftrightarrow$ are left as exercises.)

Exercise 16. Specify the semantic rules for the material conditional.

Exercise 17. Specify the semantic rules for the biconditional.

3.3.3 Entailment and validity in $\mathrm{L}_{\mathrm{Prop}}$

In Chapter 1, entailment and validity were defined in terms of cases: A entails B if and only if in any case where A is true, B is

true too, and an argument is valid if and only if in any case where all of its premises are true, its conclusion is true too. In $\mathrm{L}_{\mathrm{Prop}}$ the cases are interpretations, so these definitions can now be stated precisely.

- ϕ ENTAILS ψ, written $\phi \models \psi$, if and only if $[\![\psi]\!]^I = \mathsf{T}$ for every interpretation I such that $[\![\phi]\!]^I = \mathsf{T}$.
- More generally, $\phi_1, \ldots, \phi_n \models \psi$ if and only if $[\![\psi]\!]^I = \mathsf{T}$ for every interpretation I under which all of $\phi_1, \ldots, \phi_n$ are true.
- An argument is VALID if and only if its premises entail its conclusion. (Note that *valid* is used here of arguments; applied to a formula, it means instead that the formula is a tautology.)
- ϕ and ψ are EQUIVALENT, written $\phi \equiv \psi$, if and only if $[\![\phi]\!]^I = [\![\psi]\!]^I$ for every interpretation I.

A tautology is thus a formula entailed by every set of premises, including the empty one, and a contradiction is a formula that is not satisfiable.

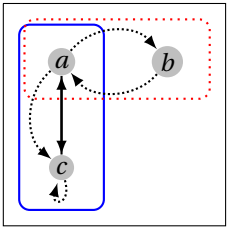
4 Predicate logic

4.1 Introduction

In Chapter 2, we gave some examples of cases, like this one:

Case 1

$M = \{a, c\}$
$F = \{a, b\}$
$R_{\mathsf{sib}} = \{\langle a, c\rangle, \langle c, a\rangle\}$
$R_{\mathsf{fan}} = \{\langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle, \langle c, c\rangle\}$



There are three individuals under consideration in this case, labeled *a*, *b*, and *c*. Let's assume that *a* can be referred to in the English language as *Alex*, *b* as *Blake*, and *c* as *Casey*. (These are names that are compatible with any gender.) Let's suppose that the English verb *likes* expresses the R_{fan} relation, and R_{sib} is the 'sibling' relation expressed by the English noun *sibling*. In English, we can state several facts about this case:

(1) a. Alex is male.
b. Alex is Casey's sibling.
c. Casey likes herself.[1]

[1] We are ignoring the semantic contribution of gender morphology in our translation to Predicate Logic, but Chapter 8 will offer a treatment of pronoun gender.

d. Alex likes someone.

These are the sorts of things that can be expressed in Predicate Logic as well.

A FORMULA is the logical equivalent of a sentence; a formula is a statement that can be true or false. In propositional logic, the smallest, most basic elements are formulas, and nothing smaller; there is nothing in propositional logic that corresponds to *Alex* or to *male*. There are many valid arguments that we would like to be able to capture that depend on our ability to break statements into smaller parts; for example, *Alex is male* entails *Someone is male*. We've seen similar arguments at the outset of the previous chapter, and we'll see plenty more as we proceed.

In Predicate Logic, there is structure within formulas. Here are formulas of Predicate Logic that can be used to express the same ideas that were expressed in English in (1), along with a suggestion for how to read them aloud:

(2) a. male(a)
'Alex is male'
b. sibling(a, c)
'Alex stands in the sibling relation to Casey'
c. likes(c, c)
'Casey likes Casey'
d. $\exists x$.likes(a, x)
'there exists an x such that Alex likes x'

Expressions of Predicate Logic are formed from a set of basic expressions, including:

- UNARY PREDICATE SYMBOLS like male
(these take one argument)
- BINARY PREDICATE SYMBOLS like sibling
(these take two arguments)
- NAMES like a and b (outside semantics and philosophy these

are more usually called INDIVIDUAL CONSTANTS, a term we also use below)

Predicate Logic is also equipped with

- VARIABLES like x and y
- QUANTIFIERS like $\exists$ ('for some') and $\forall$ ('for all')

These elementary symbols can be composed according to the syntax rules of Predicate Logic to form statements like $\mathsf{male(a)}$ and $\exists x.\mathsf{likes(a}, x)$.

Just as in Propositional Logic, formulas can be atomic, containing no other formulas, or non-atomic. Atomic formulas can be formed by putting together a unary or binary predicate together with the appropriate number of arguments (one or two, respectively), as in (2a) and (2b). More formally: There are two sorts of things that count as TERMS: variables like x and y, and individual constants like a and b.

- If π is a unary predicate and α is a term, then $\pi(\alpha)$ is a formula.
- If π is a binary predicate and α and β are terms, then $\pi(\alpha, \beta)$ is a formula.

Non-atomic formulas can be formed by putting formulas together using the rules of Propositional Logic, or with the help of quantifiers as in (2d). For example, if ϕ is a formula and ψ is a formula, then $[\phi \wedge \psi]$ is also a formula. Since $\mathsf{male(a)}$ is a formula and $\mathsf{female(a)}$ is, too, we can put these together to make a new formula: $[\mathsf{male(a)} \wedge \mathsf{female(a)}]$. Whether a given string of symbols counts as a syntactically well-formed formula in PredL is independent of whether it is true in any particular case, and even of whether or not it could be true in any case. For instance, $[\mathsf{sibling(a,b)} \wedge \neg\mathsf{sibling(a,b)}]$ is a logical contradiction (true in no model), but it is still a well-formed formula.

The existential and universal quantifiers $\exists$ and $\forall$ are used to make existential or universal formulas with the help of variables. A quantifier is always followed by a variable (the variable that it BINDS) and a formula (normally separated by a dot). For example, x is a variable, and $\mathsf{likes}(\mathsf{a}, x)$ is a formula (an atomic formula). Putting the existential quantifier $\exists$ together with x and the formula yields the new formula:

$$\exists x\,.\,\mathsf{likes}(\mathsf{a}, x)$$

This quantificational formula has three parts: the existential quantifier $\exists$, the variable x (which is said to be BOUND by the existential quantifier in this formula), and the formula $\mathsf{likes}(\mathsf{a}, x)$ (in which the variable x happens to appear).

We turn now to the official syntax of Predicate Logic (Section 4.2.1). We'll present the official semantics in Section 4.2.2. Section 4.3 gives a summary of how the language is defined.

4.2 Predicate Logic

4.2.1 Syntax

4.2.1.1 Syntax of basic expressions

The basic expressions of Predicate Logic include constants and variables. Among the constants, there are two sorts: individual constants (names), and predicates. We begin with the former.

Individual constants (names). Individual objects are named by INDIVIDUAL CONSTANTS, also known as NAMES. The names we will most typically use as examples are ones like a and b, starting with letters drawn from the beginning of the alphabet. In this book, we adopt the convention that individual constants start with a lowercase letter. In general, constants (including individual constants and predicate symbols) may contain any sequence of letters and

numbers and underscores, but no spaces. For example,

$$\mathsf{sam\_smith}$$

is a valid individual constant, but

$$\mathsf{S}$$

is not, nor is

$$\mathsf{sam\ smith}.$$

Like the variables of Predicate Logic that range over individuals in the domain, individual constants are terms, which means that they can serve as arguments to predicates.

Variables. We will allow an infinite number of variables of the form x_i, y_i, or z_i, where i is any nonnegative integer. For example, our x_i variables include x_0, x_1, x_2, and so on. In addition to these, plain x, y, and z are also variables, along with versions with one or more prime marks, such as x' ('x prime') or x'' ('x double prime').

Syntactically, variables are similar to individual constants. Both can occur in the argument position of a predicate, as in $\mathsf{likes}(x, y)$ or $\mathsf{likes}(\mathsf{a}, \mathsf{b})$ or $\mathsf{likes}(x, \mathsf{a})$. Variables that range over individuals (which is the only kind we have at present) can occur anywhere in a formula that an individual constant can occur, namely in any argument position for a predicate. That is because variables and individual constants are all TERMS. But as we will see, variables can appear in one place where quantifiers cannot, namely right after a universal or existential quantifier.

Variables can be contrasted with constants. While x is a variable, a is a constant, an individual constant in particular. Individual constants are not the only constants; unary and binary predicates fall into this category as well. The sense in which names and predicates are constants is that their reference is fixed relative to a given model, as we will see later. Variables do not have a fixed reference relative to a given model.

Predicates. True to its name, predicate logic has PREDICATES. Predicates are expressions that stand, intuitively speaking, for properties (such as being female, being Swedish, or singing) or for relations (such as loving, or being between).

The number of terms that a predicate symbol combines with is its ARITY, also called VALENCE or ADICITY (which sometimes gets misspelled as *acidity*). Unary predicates take one term, and therefore have an arity of 1. Binary predicates have an arity of 2. A TERNARY PREDICATE has an arity of 3. As an example, we might define a ternary predicate BETWEEN, and assume that it denotes the relation that holds of three objects x, y and z, if and only if x is between y and z. There is no upper limit to the arity of predicates in logic. Sometimes it is useful to regard propositional letters as ZERO-PLACE or NULLARY predicates; in particular, doing so allows the languages of predicate logic to be regarded as supersets of the languages of propositional logic. It is also common to speak of ONE-PLACE or MONADIC, TWO-PLACE or DYADIC, and generally of n-ARY, n-PLACE or n-ADIC PREDICATES.

In this book, the first letter of a predicate symbol will be lower case, as in female, swedish or sings. We will use the same style for non-unary predicates such as loves or between.

This completes our exposition of the three types of basic expressions in Predicate Logic: individual constants (names), variables, and predicates of various arities. One further symbol, the equality symbol =, is standardly set apart from the ordinary predicates, because its interpretation is fixed in advance rather than supplied by the model; we introduce it in Section 4.2.1.3 below. We turn next to the syntactic formation rules that can be used to create larger expressions.

4.2.1.2 Syntax of predication

Predicates combine with the appropriate number of terms to form ATOMIC FORMULAS. As we have seen above, male(a) is an atomic formula. Here a unary predicate male combines with a single term, enclosed in parentheses, to form an atomic formula. A binary predicate combines with two terms, enclosed in parentheses, to form an atomic formula. Thus is also an atomic formula.

The following syntactic rule, introducing predicate-argument combinations into the language, enforces a match between the arity of a predicate and the number of terms it combines with:

Syntactic rule: Predication
Given any predicate π, if n is the arity of π, and $\alpha_1, ..., \alpha_n$ is a sequence of terms, then

$$\pi(\alpha_1, ..., \alpha_n)$$

is an atomic formula.

In this rule, the symbols π and $\alpha_1, ..., \alpha_n$ are META-LANGUAGE VARIABLES (or META-VARIABLES), so u can be instantiated as any particular variable. Meta-language variables can be instantiated by a range of different particular expressions of the representation language, and they themselves do not occur in the representation language formulas. that is to say, they belong to our meta-language. (Recall that our meta-language is English with some precise set-theoretic vocabulary for describing models mixed in.)

Examples (assuming male is a unary predicate and sibling and likes are binary predicates):

- male(a)
- sibling(y, y)
- likes(x, c)

Recall that the argument to a predicate may be any term. This includes constants like a and variables like x.

4.2.1.3 Syntax of identity statements

It is useful to be able to express that two terms refer to the same individual. For this purpose, we will add a special two-place predicate to our language, the equality symbol $=$. This symbol is interpreted as the identity relation, which holds between any individual and itself, and does not hold between any distinct individuals. While identity is technically a binary predicate, it behaves differently from all other predicates in the language, and including it changes the formal properties of the resulting logic. For this reason, it is common to speak of "predicate logic with identity" or "predicate logic without identity" depending on whether the predicate is included or left out. As we will see when we turn to the semantics, $=$ is not treated as just another predicate whose interpretation varies from model to model; its interpretation is fixed as the identity relation, which is why its characteristic properties — that everything is identical to itself, and that whatever holds of an individual holds of anything identical to it — do not have to be stipulated as axioms.

Syntactically, the identity symbol is used to form atomic formulas by joining two terms. Unlike other predicate symbols, it is inserted between the terms and not in front of them.

Syntactic Rule: Identity
If α and β are terms, then $\alpha = \beta$ is an atomic formula.

By this rule, the following are all atomic formulas:

- $x = y$
- $x = \mathsf{a}$
- $\mathsf{a} = \mathsf{b}$

Note: This rule only applies to *terms*; formulas cannot be joined by an equals symbol. To join two *formulas*, the biconditional symbol $\leftrightarrow$ can be used instead.

It is common to extend predicate logic with other binary predicates taken from mathematics as well, such as $\leq$ or $\in$. These predicates often receive a special treatment, both syntactically and semantically. Syntactically, they are most commonly written in between the terms they apply to, as in $a = b$ instead of $=(a, b)$. Semantically, they are usually treated as logical rather than non-logical constants. In this chapter, the only such predicate we are adding to our logic is identity ($=$). In Chapter 9, we will add a predicate standing for the parthood relation.

4.2.1.4 Syntax of connectives

Predicate Logic has all of the connectives of Propositional Logic.

Syntactic Rule: Negation

- If ϕ is a formula, then $\neg\phi$ is a formula.

Syntactic Rule: Binary connectives
If ϕ is a formula and ψ is a formula, then so are:

- $[\phi \wedge \psi]$ 'ϕ and ψ'
- $[\phi \vee \psi]$ 'ϕ or ψ'
- $[\phi \rightarrow \psi]$ 'if ϕ then ψ'
- $[\phi \leftrightarrow \psi]$ 'ϕ if and only if ψ'

Examples:

- $\neg \text{male}(x)$

- $\neg\neg\neg \mathsf{male}(x)$
- $\neg[\mathsf{male}(x) \wedge \mathsf{female}(x)]$
- $[\neg\mathsf{male}(x) \wedge \neg\mathsf{female}(x)]$

Note: The last two will not turn out to be equivalent, given the semantic rules we define later. According to those rules, $\neg[\mathsf{male}(x) \wedge \mathsf{female}(x)]$ describes a condition that x satisfies if it is not the case that x is both male and female. $[\neg\mathsf{male}(x) \wedge \neg\mathsf{female}(x)]$ describes a condition that is satisfied by x if they are neither male nor female.

4.2.1.5 Syntax of quantifiers

The formation rules for the universal quantifier $\forall$ and the existential quantifier $\exists$ are as follows:

Syntactic rule for $\mathrm{L}_{\mathrm{Pred}}$: Quantification
Given any variable u, if ϕ is a formula, then

$$[\forall u.\phi]$$

is a formula, and so is

$$[\exists u.\phi]$$

In this rule, the symbols u and ϕ are both meta-variables, so u can be instantiated as any particular variable. Thus $[\forall x.\mathsf{happy}(x)]$ is a well-formed formula (but $[\forall u.\mathsf{happy}(u)]$ and $[\forall x.\phi]$ are not).

As an abbreviatory shorthand, whenever there is no risk of ambiguity, we may drop the brackets around the formula (as we have already done in many cases). We may also drop the dot after the variable when it is immediately followed by a bracket, e.g. $\forall x[\mathsf{happy}(x) \rightarrow \mathsf{friendly}(x)]$. In a formula of the form $[\forall u.\phi]$ or $[\exists u.\phi]$, we call ϕ the SCOPE of the quantifier. When the outer

brackets are dropped, the dot indicates that the scope of its quantifier extends as far to the right as possible.

In the formula $\mathsf{happy}(x)$, x occurs without an associated quantifier. In this sense, x occurs FREE, as opposed to BOUND by a quantifier. A formula containing free variables is called an OPEN FORMULA. A formula containing no free variables is called a CLOSED FORMULA. As a special case, this also includes a formula that contains no variables at all. A formula containing one or more free variables is called an OPEN FORMULA. A closed formula is also called a SENTENCE. The distinctions introduced in this paragraph are syntactic, rather than semantic, in the sense that they only talk about the form of the expressions. However, there are semantic consequences of this distinction, as we will see.[2]

Whether a variable is free or bound is defined syntactically as follows:

- In an atomic formula, any variable is free.
- Any free variable in ϕ is also free in $\neg\phi$. Any free variable in ϕ or ψ is free in $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \rightarrow \psi]$, and $[\phi \leftrightarrow \psi]$.
- Any free variable in ϕ is free in $[\forall u.\phi]$ and $[\exists u.\phi]$, except for u, which is bound in formulas of that form.

4.2.1.6 Exercises

Exercise 1. Consider the following formulas.

[2] An odd feature of predicate logic is that if ϕ is a closed formula, ϕ is equivalent to $[\forall u.\phi]$ as well as to $[\exists u.\phi]$. For example, $\mathsf{swedish(a)}$ is equivalent to $[\forall x.\mathsf{swedish(a)}]$ and to $[\exists x.\mathsf{swedish(a)}]$. These formulas are all true (on the intended interpretation) just in case Alex is Swedish. To put it differently, if it is true that Alex is Swedish, then it is also true of every individual, and of some individual, that Alex is Swedish.

(a) $[\mathsf{happy}(\mathsf{a}) \wedge \mathsf{happy}(\mathsf{a})]$

(b) $\mathsf{happy}(\mathsf{b})$

(c) $\mathsf{happy}(\mathsf{c},\mathsf{c})$

(d) $\neg\neg\mathsf{happy}(\mathsf{a})$

(e) $\forall x.\mathsf{happy}(x)$

(f) $\forall x.\mathsf{happy}(y)$

(g) $\exists x.\mathsf{loves}(x,x)$

(h) $\exists x.\exists z.\mathsf{loves}(x,z)$

(i) $\exists x.\mathsf{loves}(x,z)$

(j) $\exists x.\mathsf{happy}(\mathsf{m})$

Questions:

(i) Which of the above are well-formed formulas of $\mathrm{L}_{\mathrm{Pred}}$?

(ii) Of the ones that are well formed in $\mathrm{L}_{\mathrm{Pred}}$, which have free variables in them? (In other words, which of them are *open formulas*?)

Recommended: Express your answer in the form of a table, with one column for each question.

Exercise 2. Which of the following symbols (or symbol sequences) are **terms**? Recall that terms are the sorts of things that can serve as the argument to a predicate.

(i) x

(ii) a

(iii) $\mathsf{sibling}(x)$

(iv) male

(v) $\neg$

(vi) $\forall$

(vii) c

(viii) z

(ix) $\mathsf{sibling}(x,\mathsf{a})$

(x) $x \wedge y$

Exercise 3. Which of the following symbol sequences are well-formed according to the syntax of Predicate Logic? Assume that sibling and likes are binary predicates, male and female are unary predicates, and a, b, and c are individual constants. Letters drawn from the end of the alphabet such as x, y, and z are variables.

(i) $\mathsf{likes}(\mathsf{a},\mathsf{a})$

(ii) $\mathsf{siblings}(\mathsf{a} \wedge \mathsf{b})$

(iii) $[\mathsf{sibling}(\mathsf{a},\mathsf{b}) \wedge \mathsf{sibling}(\mathsf{b},\mathsf{a})]$

(iv) $\neg\neg\neg\mathsf{male}(\mathsf{a})$

(v) $\forall x.\mathsf{sibling}(x,x)$

(vi) $\exists \mathsf{b}.\mathsf{sibling}(x,\mathsf{b})$

(vii) $\exists[\mathsf{male}(x).\mathsf{sibling}(x,\mathsf{a})]$

(viii) $\exists x[\mathsf{male}(x) \wedge \mathsf{sibling}(x,\mathsf{a})]$

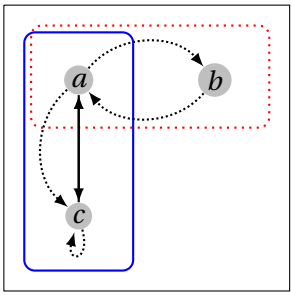
(ix) $\exists x.\forall x.\mathsf{likes}(z,\mathsf{c})$

(x) $\exists \forall x.\mathsf{likes}(x,\mathsf{c})$

4.2.2 Semantics

Whether or not a formula is *true* relative to a given case is determined by the *semantic* rules for the logic. In predicate logic (also known as first-order logic), truth is evaluated relative to a FIRST-ORDER MODEL. A first-order model is a way of organizing the information about a case like Case 1.

One feature of a first-order model is its UNIVERSE OF DISCOURSE



$D_1 = \{a, b, c\}$
$I_1(\mathsf{a}) = a$
$I_1(\mathsf{b}) = b$
$I_1(\mathsf{c}) = c$
$I_1(\mathsf{male}) = \{a, c\}$
$I_1(\mathsf{female}) = \{a, b\}$
$I_1(\mathsf{sibling}) = \{\langle a, c\rangle, \langle c, a\rangle\}$
$I_1(\mathsf{likes}) = \{\langle a, b\rangle, \langle a, c\rangle, \langle c, c\rangle\}$

Figure 4.1: $M_1 = \langle D_1, I_1\rangle$

or DOMAIN.[3] In Case 1, the domain is the set $\{a, b, c\}$; these are the individuals presumed to exist. (Note that they need not all have names.) A model also specifies a so-called INTERPRETATION for the unary and binary predicates of the language like male and sibling. Unary predicates are interpreted as sets (subsets of the model's domain); for example, male could be interpreted as the set $\{a, c\}$. Binary predicates are interpreted as relations, like the ones depicted with arrows in the diagram for Case 1.

A model for Predicate Logic comprises two parts: a domain D, specifying a non-empty set of individuals, and an interpretation function I, which assigns a value to every non-logical constant of the language. More formally, a model M is a pair $\langle D, I\rangle$, where D is a non-empty set of individuals, and I is a function whose domain is the set of non-logical constants. I maps individual constants to particular elements of D, unary predicates to subsets of D; and binary predicates to relations on $D \times D$.

To illustrate, let us define a model M_1 corresponding to Case 1 as illustrated above. We define $M_1 = \langle D_1, I_1\rangle$, defined as in Figure 4.1.

[3] This sense of the term *domain* is distinct from the sense introduced in Chapter 2, in which functions are mappings from a domain to a codomain.

4.2.2.1 Semantics of non-logical constants

Names like a and b pick out a fixed member of the domain, and their interpretation is specified by a model. For instance, a model may specify that the name a is interpreted as the individual a.

More formally, the denotation of a name like a relative to a given model M is determined by M's interpretation function I. Thus if $M = \langle D, I \rangle$ then $[\![\mathsf{a}]\!]^M = I(\mathsf{a})$. Taking our example M_1:

$$[\![\mathsf{a}]\!]^{M_1} = I_1(\mathsf{a}) = a$$

Similarly:

$$[\![\mathsf{male}]\!]^{M_1} = I_1(\mathsf{male}) = \{a, c\}$$

The double brackets $[\![\cdot]\!]$ are called DENOTATION BRACKETS, or SEMANTIC BRACKETS, and they represent a DENOTATION FUNCTION, which maps expressions of Predicate Logic to their denotations relative to a model.

Notice that in this book, denotation brackets go around expressions of the representation language. In the research literature in formal semantics, it is common to see denotation brackets around expressions of the natural language, such as English. This is one point where direct and indirect interpretation styles differ. Since we are doing indirect interpretation in this book, we will (eventually) specify translations from English to expressions of our logic, and then let the denotations of the English expressions be inherited from their translations into the logical representation language.

The quantifiers $\forall$ and $\exists$, along with symbols from propositional logic like $\neg$, $\wedge$ and $\vee$ are also constants, but they are constants of a different sort. Their interpretation does not come from a model, but from the semantic rules of the logic. These constants are called LOGICAL CONSTANTS, while the names and predicates are called NON-LOGICAL CONSTANTS. Logical and non-logical constants are treated differently by the semantic rules for Predicate Logic. Non-logical constants depend on a model for their interpretation, while logical constants are model-independent.

Semantic Rule: Non-logical constants
If α is a non-logical constant and $M = \langle D, I \rangle$, then:

$$[\![\alpha]\!]^M = I(\alpha)$$

4.2.2.2 Semantics of predication

The truth or falsity of a formula in Predicate Logic is evaluated relative to a model like this based on semantic rules. 'Being true' (in a model M) is implemented as denoting the truth value T (relative to M). There are different semantic rules for different sorts of formulas.

The truth of an atomic formula with a unary predicate relative to a given model M is determined as follows. If π is a unary predicate such as male and α is a term such as the individual constant a or the variable x, then:

$$[\![\pi(\alpha)]\!]^M = \mathsf{T} \text{ if } [\![\alpha]\!]^M \in [\![\pi]\!]^M, \text{ and } \mathsf{F} \text{ otherwise.}$$

(Here, π and α are meta-language variables, variables ranging over expressions of the representation language.) This says that $\pi(\alpha)$ is true relative to M if the denotation of α in M is a member of the set denoted by π in M.

The careful reader may already be able to guess how the semantic rule for atomic formulas involving binary predicates will be defined. Here is a hint. It follows from the definition that:

$$[\![\mathsf{sibling(a,c)}]\!]^M = \mathsf{T}$$

if and only if the pair $\langle [\![\mathsf{a}]\!]^M, [\![\mathsf{c}]\!]^M \rangle$ is a member of the relation denoted by sibling in M. (Recall that relations are sets of pairs.) Take the case of $M_1 = \langle D_1, I_1 \rangle$, where $I_1(\mathsf{sibling}) = \{\langle a, c \rangle, \langle c, a \rangle\}$. $[\![\mathsf{a}]\!]^{M_1} = a$ and $[\![\mathsf{c}]\!]^{M_1} = c$. Since $\langle a, c \rangle \in \{\langle a, c \rangle, \langle c, a \rangle\}$, the formula is true in M_1. In general, if π is a binary predicate, then:

$$[\![\pi(\alpha, \beta)]\!]^M = \mathsf{T} \text{ if } \langle [\![\alpha]\!]^M, [\![\beta]\!]^M \rangle \in [\![\pi]\!]^M, \text{ and } \mathsf{F} \text{ otherwise.}$$

The following rule is intended to handle predicates of any arity, including the unary and binary cases we have just seen, as well as predicates of higher arities: ternary, quaternary, etc.

Semantic Rule: Predication
If π is a predicate of arity n and $\alpha_1, ..., \alpha_n$is a sequence of terms, then:

$[\![\pi(\alpha_1, ...\alpha_n)]\!]^M = \mathsf{T}$ if $\langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M \rangle \in [\![\pi]\!]^M$, and F otherwise.

To make sure this works as expected in the unary case, we adopt the convention that $\langle [\![\alpha_n]\!]^M \rangle = [\![\alpha_n]\!]^M$.

4.2.2.3 Semantics of identity statements

Unlike binary predicate constants, the interpretation of identity is independent of the model. This makes identity a logical rather than non-logical constant.

Semantic Rule: Identity
If α and β are terms, then $[\![\alpha = \beta]\!]^M = \mathsf{T}$ if $[\![\alpha]\!]^M = [\![\beta]\!]^M$, and F otherwise.

With the use of the equals sign (=) in the meta-language, we mean to invoke a relation that necessarily holds only between an object and itself. This concept is known in the philosophical literature as NUMERICAL IDENTITY.

4.2.2.4 Semantics of connectives

The semantics of the connectives are familiar from Propositional Logic.

Semantic Rule: Connectives

- $[\![\phi \wedge \psi]\!]^M = \mathsf{T}$ if $[\![\phi]\!]^M = \mathsf{T}$ and $[\![\psi]\!]^M = \mathsf{T}$, and F otherwise.
- $[\![\phi \vee \psi]\!]^M = \mathsf{T}$ if $[\![\phi]\!]^M = \mathsf{T}$ or $[\![\psi]\!]^M = \mathsf{T}$, and F otherwise.
- $[\![\phi \rightarrow \psi]\!]^M = \mathsf{T}$ if $[\![\phi]\!]^M = F$ or $[\![\psi]\!]^M = \mathsf{T}$, and F otherwise.
- $[\![\phi \leftrightarrow \psi]\!]^M = \mathsf{T}$ if $[\![\phi]\!]^M = [\![\psi]\!]^M$, and F otherwise.

4.2.2.5 Variables and assignment functions

Recall this argument from the beginning of Chapter 3, which we were not able to represent in propositional logic:

(3) Aristotle taught Alexander the Great.
Alexander the Great was a king.
∴ Aristotle taught a king.

Construing teaching as a binary relation that holds between teachers and their students, the conclusion of this argument is true in any case where Aristotle stands in the teaching relation to an individual that is a king. How can we express this formally? If we had names for all of the kings, then we could express this using the tools we have by saying something along the lines of, "Aristotle taught King So-and-So or Aristotle taught King Such-and-Such or ..." and so on for all of the kings. But this is quite inconvenient, and it will not work if there are individuals without names. All we want to say is that there is some entity, call it x, such that Aristotle taught x and x is a king. This can be done using variables. Vari-

ables do not have a fixed interpretation relative to a given model; they do not pick out any particular member of the domain.

The semantics of quantified formulas can be thought of in terms of SATISFACTION. The condition that the object should satisfy may be written as follows:

(4) $[\mathsf{taught}(\mathsf{aristotle}, x) \wedge \mathsf{king}(x)]$

This is a well-formed formula of first-order logic, but it does not make a claim, even once the model is fixed; it just describes a condition that whatever individual x stands for might or might not satisfy. This is because the occurrence of the variable x in this formula is not BOUND by any quantifier (so it is FREE). By an OCCURRENCE of a variable we mean a particular place where the variable appears; a single variable may occur several times in one formula, and each occurrence may be bound or free independently of the others. To make the claim that there is some individual that satisfies this condition, we may use the EXISTENTIAL QUANTIFIER, $\exists$.

(5) $\exists x[\mathsf{taught}(\mathsf{aristotle}, x) \wedge \mathsf{king}(x)]$

This can be read, "There is (or: exists) an x such that Aristotle taught x and x is a king" or "For some x, Aristotle taught x and x is a king". And this formula will be true in any model where the individual denoted by aristotle stands in the relation denoted by taught to some element of the set denoted by king, or as we will put it, in any model where there is a king that Aristotle taught (we will use similar simplifications from now on).

The other quantifier of predicate logic is the UNIVERSAL QUANTIFIER, written $\forall$. If we had used the universal quantifier instead of the existential quantifier in the formula in (5), we would have expressed the claim that *everything* satisfies the condition in (4). Thus everything was taught by Aristotle and everything is a king. That is probably not something one would ever feel the urge to express, but there are plenty of other practical uses for the universal quantifier. For example, consider the sentence *Every philosopher*

studies Aristotle. We can represent this as follows:

(6) $\forall x[\text{philosopher}(x) \rightarrow \text{studies}(x, \text{aristotle})]$

This can be read, "For all x, if x is a philosopher, then x studies Aristotle." ("For every x" is also fine instead of "For all x", and we will use both formulations interchangeably.) We would be saying something very different if we had a conjunction symbol ($\wedge$) instead of a material conditional arrow ($\rightarrow$) in this formula, thus:

(7) $\forall x[\text{philosopher}(x) \wedge \text{studies}(x, \text{aristotle})]$

This says, "For all x, x is a philosopher and x studies Aristotle" – in other words, "Everything/everyone is a philosopher and everything/everyone studies Aristotle."

Let us take some time to reflect on why the universally quantified formula with the material conditional above expresses the *every* claim, that every philosopher studies Aristotle. We will formalize the semantics of universally quantified statements shortly, but intuitively, here is how it works. What this formula expresses is that each element of the domain satisfies the condition:

(8) $[\text{philosopher}(x) \rightarrow \text{studies}(x, \text{aristotle})]$

As we will see, the semantics for $\forall$ asks us to go through each individual in the domain, and consider what happens when x is interpreted as that individual. There are two types of cases that are important to consider: the value of x is a philosopher, or the value of x is not a philosopher. Consider a value for x which is not a philosopher. For this value of x, the condition

$$\text{philosopher}(x)$$

is not met, so the antecedent is false. By the definition of the material conditional, this means that the conditional as a whole is true. So any value for x that is not a philosopher vacuously satisfies $[\text{philosopher}(x) \rightarrow \text{studies}(x, \text{aristotle})]$. The only kind of value for

x that could fail to satisfy this condition would be a philosopher that did not study Aristotle. Then the antecedent would be true, and the consequent would be false, so the conditional statement as a whole would be false. If there are no philosophers that do not study Aristotle, then the formula is true. That is close to what *Every philosopher studies Aristotle* says, but not exactly: the formula also comes out true when there are no philosophers at all, whereas an utterance of the English sentence would normally convey that there are some. We return to this point in Chapter 8, where *every* is given a presupposition that its restrictor is non-empty.

Exercise 4. Which of the following statements in first-order logic better represents the denotation of *Every cellist smokes*?

(a) $\forall x[\mathsf{cellist}(x) \rightarrow \mathsf{smokes}(x)]$

(b) $\forall x[\mathsf{cellist}(x) \wedge \mathsf{smokes}(x)]$

On the other hand, the $\rightarrow$ has a very counterintuitive interaction with the existential quantifier. The conjunction symbol in combination with an existential quantifier can be used to state the existence of an individual satisfying two properties. For example, $\exists x[\mathsf{pirate}(x) \wedge \mathsf{sailor}(x)]$ means that the intersection between pirates and sailors is non-empty. This is a good representation of the meaning of *Some pirate is a sailor*. By contrast, $\exists x[\mathsf{pirate}(x) \rightarrow \mathsf{sailor}(x)]$ is true as long as there is at least one non-pirate, because a material conditional is true whenever the antecedent is false. It doesn't matter whether that non-pirate is a sailor or not. So this does not capture the meaning of *Some pirate is a sailor*.

Exercise 5. Which of the following statements in first-order logic better represents the denotation of *Some cellist smokes*?

(a) $\exists x[\mathsf{cellist}(x) \rightarrow \mathsf{smokes}(x)]$

(b) $\exists x[\mathsf{cellist}(x) \wedge \mathsf{smokes}(x)]$

Now consider the following formula:

(9) $\forall x[\mathsf{linguist}(x) \rightarrow \exists y[\mathsf{philosopher}(y) \wedge \mathsf{admires}(x,y)]]$

If we were to read this aloud, symbol for symbol, we would say, "For every x, if x is a linguist, then there exists a y such that y is a philosopher and x admires y." A more natural way of putting this would be "Every linguist admires a philosopher." But "Every linguist admires a philosopher" is actually ambiguous. It could mean two things:

1. For every linguist, there is some philosopher that the linguist admires (possibly a different philosopher for every linguist).

2. There is one lucky philosopher such that every linguist admires that philosopher.

The latter reading can be translated as follows:

(10) $\exists y[\mathsf{philosopher}(y) \wedge \forall x[\mathsf{linguist}(x) \rightarrow \mathsf{admires}(x,y)]]$

Predicate logic is thus a tool for teasing apart these kinds of ambiguities in natural language. What we have just seen is an instance of QUANTIFIER SCOPE AMBIGUITY. The first reading is the one where "every linguist" takes WIDE SCOPE over "a philosopher". On the second reading, "every linguist" has NARROW SCOPE with respect to "a philosopher".

Quantifiers can also take wide or narrow scope with respect to negation. Consider the sentence "Everybody isn't happy". This could mean either one of the following:

(11) a. $\forall x.\neg \mathsf{happy}(x)$
b. $\neg\forall x.\mathsf{happy}(x)$

The first formula, where the universal quantifier takes wide scope over negation says, "For every x, it is not the case that x is happy." The second formula, where the quantifier has narrow scope with respect to negation says, "It is not the case that for every x, x is happy." The first one states that nobody is happy. The second one states merely that there is at least one person who is not happy.

Now for the semantics. We continue to treat models as pairs consisting of a domain and an interpretation function, so a given model M will be defined as $\langle D, I\rangle$ where D is the set of individuals in the domain of the model, and I is a function giving a value to every non-logical constant in the language. Informally,

(12) $\forall x.\mathsf{happy}(x)$

is true in a model M if (and only if) no matter which individual we assign as the interpretation of x,

(13) $\mathsf{happy}(x)$

is true. Likewise, informally,

(14) $\exists x.\mathsf{happy}(x)$

is true iff we can find *some* individual to assign to x that makes $\mathsf{happy}(x)$ true.

Since we are doing first-order logic, all our variables range over individuals. In higher-order logic, variables can also stand for predicates. Here are two examples of statements that can be expressed as a single formula in higher-order logic but not in first-order logic:

(15) a. Napoleon had all the properties of a good general.
b. No two distinct objects have the same properties.

Example (15b) is often referred to as the law of the Identity of In-

discernibles, or Leibniz's Law. We will put off higher-order logic until Chapter 5.

As we have seen, a formula can in principle have multiple quantifiers. For example:

$$\forall x[\mathsf{happy}(x) \rightarrow \exists y.\mathsf{likes}(x,y)]$$

This says, 'everything that is happy likes something.' Whether or not it is true, it contains two variables and two quantifiers. The outermost formula is true if every individual in the domain is an x such that:

$$[\mathsf{happy}(x) \rightarrow \exists y.\mathsf{likes}(x,y)]$$

In order to evaluate whether this holds for a given x that is happy, we will need to determine whether there is a y that x likes. So we will need to hold the value of x fixed while we look for a suitable y. For sentences with multiple quantifiers, then, we need to simultaneously consider the values we are assigning to multiple variables. ASSIGNMENT FUNCTIONS allow us to do just that.

In order to interpret an open formula we need not only a model, but also an ASSIGNMENT FUNCTION alongside it. An assignment function is a function that specifies for each variable, how that variable is to be interpreted, by mapping it to an individual. Here are some examples of assignment functions:

$$g_1 = \begin{bmatrix} x & \rightarrow & \text{Alex} \\ y & \rightarrow & \text{Blake} \\ z & \rightarrow & \text{Blake} \\ \ldots & & \end{bmatrix} \qquad g_2 = \begin{bmatrix} x & \rightarrow & \text{Blake} \\ y & \rightarrow & \text{Casey} \\ z & \rightarrow & \text{Blake} \\ \ldots & & \end{bmatrix}$$

The domain of an assignment function is the set of variables.

Thus the interpretation of formulas in Predicate Logic depends on two sources of information: the model, and the assignment function. The model tells us, for example, who is happy, and the assignment function determines a value for, say, x. For uniformity, our denotation function will always be relativized to both a

model and an assignment function, although sometimes the assignment function will not make a difference to the denotation. We typically use the letter g to stand for an assignment function, so instead of

$$[\![\phi]\!]^{M}$$

we will now write:

$$[\![\phi]\!]^{M,g}$$

where g stands for an assignment function. The *denotation of the variable u with respect to model M and assignment function g*, written:

$$[\![u]\!]^{M,g}$$

is simply whatever g maps u to. We can express this more formally as follows:

Semantic rule for $\mathrm{L}_{\mathrm{Pred}}$: Variables
$[\![u]\!]^{M,g} = g(u)$

For example, $[\![x]\!]^{M,g_1} = g_1(x)$ = Alex, and $[\![x]\!]^{M,g_2} = g_2(x)$ = Blake (for any model M that contains Blake).

Exercise 6. In this exercise, use the assignment functions g_1 and g_2 that we defined above.

(a) What is $g_1(y)$?

(b) What is $[\![y]\!]^{M,g_1}$ (for any model M)?

(c) What is $g_2(y)$?

(d) What is $[\![y]\!]^{M,g_2}$ (for any model M)?

From now on, our semantic denotation brackets will have two superscripts: one for the model, and one for the assignment function. As a reminder, the model is just a pair consisting of a domain (which consists of all things that can potentially occur as denotations of predicates, of individuals, of functions, etc.) and an interpretation function (which applies to non-logical constants in the language). The assignment function applies to variables in the language, and is not part of the model. In some cases, the choice of assignment function will not make any difference for the semantic value of the expression. For example, take any model M in which the constant happy is defined. $[\![\text{happy}]\!]^{M,g_1}$ will be the same as $[\![\text{happy}]\!]^{M,g_2}$ for any two assignments g_1 and g_2, because happy is a constant. Since it is a non-logical constant, its semantic value depends on the model, but that is the only thing that it depends on. In particular, it does not depend on any assignment function. But the value of the formula

$$\text{happy}(x)$$

depends on the value that is assigned to x. Whether $\text{happy}(x)$ is true or not depends on how x is interpreted, and this is given by the assignment function.

The choice of assignment function doesn't always make a difference for the interpretation of an expression. It only makes a difference when the formula contains free variables. For example, in the formula

$$\text{happy}(x)$$

the variable x is not bound by any quantifier (so it is a free variable). So the semantic value of this formula relative to M and g depends on what g assigns to x. In contrast, a closed formula such as $\forall x.\text{happy}(x)$ has the same value relative to every assignment function.

4.2.2.6 Semantics of quantifiers

Now let us consider the formula $\exists x.\mathsf{happy}(x)$. This is true if we can find one individual to assign x to such that $\mathsf{happy}(x)$ is true. Suppose we are trying to determine whether $\exists x.\mathsf{happy}(x)$ is true with respect to a given model M and an assignment function g. We can show that the formula is true if we can find a variant of g on which the variable x is assigned to some happy individual.

Let us use the expression

$$g[x \mapsto \text{Frida}]$$

to describe an assignment function which differs from g, if at all, only in that $g(x) = \text{Frida}$. That is to say, $g[x \mapsto \text{Frida}]$ is like g except that it maps x to Frida while g itself may or may not do so. If g already happens to map x to Frida, then $g[x \mapsto \text{Frida}]$ is exactly the same as g; otherwise, the two functions differ when it comes to the value of x, and are otherwise the same.

In general, for any variable u and any individual k,

$$g[u \mapsto k]$$

is an assignment function that is exactly like g with the possible exception that the value of $g(u)$ is k. Here, k is a symbol of our meta-language that stands for an individual in the domain, and u is a meta-variable over variables—that is to say, u is a symbol of our meta-language that stands for a variable of our object language. We call this a u-VARIANT OF g. If g already maps u to k then, $g[u \mapsto k]$ is the same as g. This technique lets us keep everything the same in g except for the variable of interest.

Let us consider an example using a particular assignment function, g_1 from above:

$$g_1 = \begin{bmatrix} x & \rightarrow & \text{Alex} \\ y & \rightarrow & \text{Blake} \\ z & \rightarrow & \text{Blake} \\ \dots & & \end{bmatrix}$$

$g_1[y \mapsto \text{Casey}]$ would be as follows:

$$g_1[y \mapsto \text{Casey}] = \begin{bmatrix} x & \rightarrow & \text{Alex} \\ y & \rightarrow & \text{Casey} \\ z & \rightarrow & \text{Blake} \\ \ldots & & \end{bmatrix}$$

We changed it so that y maps to Casey and kept everything else the same.

Exercise 7. (a) What is $g_1[z \mapsto \text{Casey}](x)$? (I.e., what does $g_1[z \mapsto \text{Casey}]$ assign to x?)

(b) What is $g_1[z \mapsto \text{Casey}](y)$?

(c) What is $g_1[z \mapsto \text{Casey}](z)$?

The rule for existential quantification to be presented very shortly derives the following semantics for $\exists x.\mathsf{happy}(x)$:

$[\![\exists x.\mathsf{happy}(x)]\!]^{M,g} = \mathsf{T}$ iff there is an individual $k \in D$ such that:
$[\![\mathsf{happy}(x)]\!]^{M,g[x \mapsto k]} = \mathsf{T}$.

What this says is that given a model M and an assignment function g, the sentence $\exists x.\mathsf{happy}(x)$ is true with respect to M and g if we can *modify* the assignment function g in such a way that x has a denotation that makes $\mathsf{happy}(x)$ true. In general:

Semantic Rule: Existential quantification
$[\![\exists x.\phi]\!]^{M,g} = \mathsf{T}$ iff there is an individual $k \in D$ such that:

$$[\![\phi]\!]^{M,g[x \mapsto k]} = \mathsf{T}$$

Universal quantification imposes more stringent requirements than existential quantification. To show that the formula $\forall x.\mathsf{happy}(x)$ was true, we would have to consider assignments of x to every element of the domain, not just one. (To show that it is false is easier; then you just have to find one unhappy individual.) If $\mathsf{happy}(x)$ turns out to be true no matter what the assignment function maps x to, then $\forall x.\mathsf{happy}(x)$ is true. Otherwise it is false. So the official semantics of the universal quantifier is as follows:

Semantic Rule: Universal quantification
$[\![\forall v.\phi]\!]^{M,g} = \mathsf{T}$ iff for all individuals $k \in D$:

$$[\![\phi]\!]^{M,g[v\mapsto k]} = \mathsf{T}$$

One important feature of the semantics for quantifiers and variables in first-order logic using assignment functions is that it scales up to formulas with multiple quantifiers. Recall the quantifier scope ambiguity in *Every linguist admires a philosopher* that we discussed at the beginning of the section. That sentence was said to have two readings, which can be represented as follows:

$$\forall x[\mathsf{linguist}(x) \rightarrow \exists y[\mathsf{philosopher}(y) \wedge \mathsf{admires}(x,y)]]$$

$$\exists y[\mathsf{philosopher}(y) \wedge \forall x[\mathsf{linguist}(x) \rightarrow \mathsf{admires}(x,y)]]$$

We will spare you a step-by-step computation of the semantic value for these sentences in a given model. We will just point out that in order to verify the first kind of sentence, with a universal quantifier outscoping an existential quantifier, one would consider modifications of the input assignment for every member of the domain, and within that, try to find modifications of the modified assignment for some element of the domain making the existential statement true. To verify the second kind of sentence, one would try to find a single modification of the input assignment for

the outer quantifier (the existential quantifier), such that modifications of that modified assignment for every member of the domain verify the embedded universal statement. This procedure will work for indefinitely many quantifiers.

4.2.2.7 Exercises

Exercise 8. The following is a list of sentences (A.-D.) that sometimes appear as one of the corners of the square of opposition. For each one, choose the formula in Predicate Logic (i.-vii.) that best captures its truth conditions.

A. Every sailor is a pirate.

B. Some sailor is a pirate.

C. No sailor is a pirate.

D. Some sailor is not a pirate.

Formulas:

i. $\neg\forall x[\mathsf{sailor}(x) \wedge \mathsf{pirate}(x)]$

ii. $\exists x[\mathsf{sailor}(x) \wedge \mathsf{pirate}(x)]$

iii. $\neg\exists x[\mathsf{sailor}(x) \wedge \mathsf{pirate}(x)]$

iv. $\forall x[\mathsf{sailor}(x) \rightarrow \mathsf{pirate}(x)]$

v. $\forall x[\mathsf{sailor}(x) \wedge \mathsf{pirate}(x)]$

vi. $\exists x[\mathsf{sailor}(x) \rightarrow \mathsf{pirate}(x)]$

vii. $\neg\exists x[\mathsf{sailor}(x) \rightarrow \mathsf{pirate}(x)]$

viii. $\exists x[\mathsf{sailor}(x) \wedge \neg\mathsf{pirate}(x)]$

Not every formula will be used.

Exercise 9. Match each formula (A.-D.) in Predicate Logic to an equivalent (i.-iv.) one (one that has the same truth conditions).

A. $\forall x[P(x) \rightarrow Q(x)]$

B. $\neg\exists x[P(x) \wedge Q(x)]$

C. $\exists x[P(x) \wedge \neg Q(x)]$

D. $\exists x[P(x) \rightarrow Q(x)]$

i. $\exists x[\neg P(x) \vee Q(x)]$

ii. $\forall x[\neg Q(x) \rightarrow \neg P(x)]$

iii. $\neg\forall x[P(x) \rightarrow Q(x)]$

iv. $\forall x[P(x) \rightarrow \neg Q(x)]$

Exercise 10. Match each English sentence (A.-J.) below to a formula of Predicate Logic that shares its truth conditions.

Assume that the proper name *Alex* is translated into Predicate Logic as the individual constant a, *Blake* as b, and *Casey* as c. (These are names that are not restricted to any gender.)

Sentences of English:

A. Everyone who has a sibling is male.

B. Everyone has a brother.

C. Nobody has a brother they like.

D. Nobody has a brother who likes them.

E. Blake is Casey's only sibling.

F. Blake has only Casey as a sibling.

G. Blake and Casey are each other's sisters.

H. Blake and Casey are both sisters.

I. Every man likes a woman who likes him.*

J. Every man is liked by a woman.

*Based on an example from Montague's "Proper Treatment of Quantification in Ordinary English".

Assume *man* is translated using male.

Formulas of Predicate Logic:

i. $[\text{sibling}(\text{c},\text{b}) \wedge \neg\exists x[x \neq \text{c} \wedge \text{sibling}(x,\text{b})]]$

ii. $\forall x[\exists y \text{sibling}(y,x) \rightarrow \text{male}(x)]$

iii. $\forall x[\text{male}(x) \rightarrow \exists y[\text{female}(y) \wedge \text{likes}(y,x)]]$

iv. $[\exists x[\text{sibling}(\text{b},x) \wedge \text{female}(\text{b})] \wedge \exists x[\text{sibling}(\text{c},x) \wedge \text{female}(\text{c})]]$

v. $[\text{sibling}(\text{b},\text{c}) \wedge \neg\exists x[x \neq \text{b} \wedge \text{sibling}(x,\text{c})]$

vi. $\forall x[\text{male}(x) \rightarrow \exists y[\text{likes}(x,y) \wedge [\text{female}(y) \wedge \text{likes}(y,x)]]]$

vii. $\forall x \exists y[\text{sibling}(y,x) \wedge \text{male}(y)]$

viii. $[[\text{sibling}(\text{b},\text{c}) \wedge \text{female}(\text{b})] \wedge [\text{sibling}(\text{c},\text{b}) \wedge \text{female}(\text{c})]]$

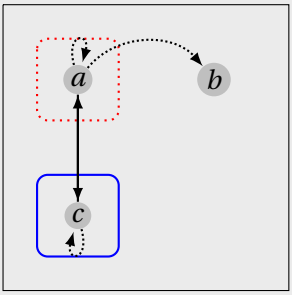
ix. $\neg\exists x \exists y[[\text{sibling}(y,x) \wedge \text{male}(y)] \wedge \text{likes}(y,x)]$

x. $\neg\exists x \exists y[[\text{sibling}(y,x) \wedge \text{male}(y)] \wedge \text{likes}(x,y)]$

Note: These formulas don't distinguish between entailed and presupposed content; we'll get to how to represent presupposition in logic later.

Exercise 11. Let us define a particular model $M_1 = \langle D_1, I_1 \rangle$ where $D_1 = \{a, b, c\}$, for individuals a, b, and c. Assume that a and c are

siblings, a is female, c is male, and b is neither, and a likes herself and b, and c likes himself, and nobody likes anybody else.



$$
\begin{aligned}
&D_1 = \{a, b, c\}\\
&I_1(\mathsf{a}) = a\\
&I_1(\mathsf{b}) = b\\
&I_1(\mathsf{c}) = c\\
&I_1(\mathsf{male}) = \{c\}\\
&I_1(\mathsf{female}) = \{a\}\\
&I_1(\mathsf{sibling}) = \{\langle a, c\rangle, \langle c, a\rangle\}\\
&I_1(\mathsf{likes}) = \{\langle a, a\rangle, \langle a, b\rangle, \langle c, c\rangle\}
\end{aligned}
$$

For which of the following formulas ϕ is it the case that $[\![\phi]\!]^{M_1} = T$? In other words, which of the following formulas are true in M_1?

(i) $\exists x.\mathsf{likes}(x, x)$

(ii) $\forall x[\mathsf{likes}(x, \mathsf{a}) \to \mathsf{female}(x)]$

(iii) $\forall x[\mathsf{likes}(x, \mathsf{a}) \wedge \mathsf{female}(\mathsf{a})]$

(iv) $\exists x.\forall y[\mathsf{male}(y) \to \mathsf{likes}(y, x)]$

(v) $\forall y \exists x[\mathsf{male}(x) \wedge \mathsf{likes}(x, y)]$

(vi) $\forall y \exists x.\mathsf{sibling}(y, x)$

(vii) $\forall y \exists x.\mathsf{likes}(y, x)$

(viii) $\forall y \exists x.\mathsf{likes}(x, y)$

(ix) $\forall y[\mathsf{likes}(y, y) \to \mathsf{male}(y)]$

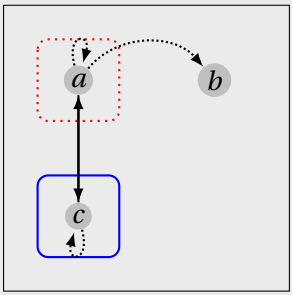
(x) $\neg \exists x.\forall y.\mathsf{sibling}(x, y)$

Exercise 12. In this question, we will consider **open formulas**, containing variables that are not bound by any quantifier. The truth value of an open formula depends on an **assignment function**, which assigns a value to every variable.

Let us define a particular assignment function g_0 as follows:

$$g_0 = \begin{bmatrix} x \to a \\ y \to a \\ z \to c \end{bmatrix}$$

Assume M_0 is still as follows:



Based on the semantic rules for Predicate Logic as defined in this chapter, which of the following formulas are true relative to M_0 and g_0? In other words, for which of the following expressions ϕ is it the case that $[\![\phi]\!]^{M_0,g_0} = T$?

(i) $\mathsf{likes}(x, \mathsf{b})$

(ii) $\mathsf{likes}(z, z)$

(iii) $\mathsf{sibling}(x, z)$

(iv) $\mathsf{likes}(\mathsf{b}, x)$

(v) $\forall y.\mathsf{likes}(x, y)$
N.b. that the universal quantifier binds y!

(vi) $\exists x.\mathsf{likes}(z,x)$
Again, notice that a quantifier has bound x!

(vii) $\exists x[\mathsf{sibling}(y,x) \wedge \mathsf{male}(x)]$

(viii) $[\mathsf{likes}(y,x) \wedge \mathsf{male}(x)]$

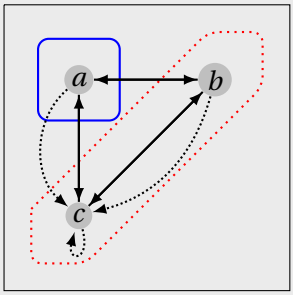
(ix) $\mathsf{likes}(y,x)$

(x) $\exists z[\mathsf{male}(z) \wedge \mathsf{likes}(\mathsf{c},z)]$

Exercise 13. Here is a formula of Predicate Logic:

$$\forall x.\mathsf{likes}(x,y)$$

Here is a model, called M_1:



$M_1 = \langle D, I_1 \rangle$
$D = \{a,b,c\}$
$I_1(\mathsf{a}) = a$; $I_1(\mathsf{b}) = b$; $I_1(\mathsf{c}) = c$
$I_1(\mathsf{male}) = \{a\}$
$I_1(\mathsf{female}) = \{b,c\}$
$I_1(\mathsf{likes}) = \{\langle a,c\rangle, \langle b,c\rangle, \langle c,c\rangle\}$
$I_1(\mathsf{sibling}) = \{\langle a,b\rangle, \langle b,a\rangle, \langle a,c\rangle, \langle c,a\rangle, \langle b,c\rangle, \langle c,b\rangle\}$

Here are four assignment functions:

$$g_1 = \begin{bmatrix} x \to a \\ y \to a \\ z \to c \end{bmatrix} \quad g_2 = \begin{bmatrix} x \to c \\ y \to c \\ z \to c \end{bmatrix}$$

$$g_3 = \begin{bmatrix} x \to b \\ y \to c \\ z \to a \end{bmatrix} \quad g_4 = \begin{bmatrix} x \to a \\ y \to b \\ z \to c \end{bmatrix}$$

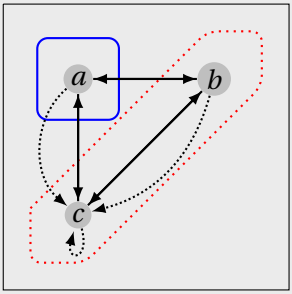
Which of these assignment functions make(s) the formula true relative to M_1?

Exercise 14. Here is a formula:

$$\forall x.\forall y[[\mathsf{sibling}(x,y)\wedge\mathsf{female}(y)]\rightarrow\neg\mathsf{likes}(x,y)]$$

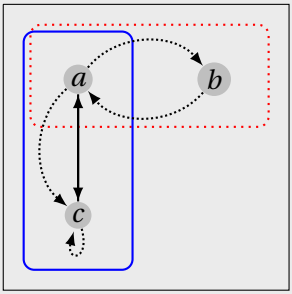
In which of the cases below is this formula true?

(i)



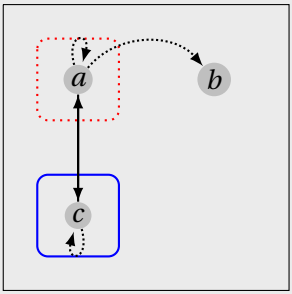
$M_1 = \langle D, I_1\rangle$
$D = \{a, b, c\}$
$I_1(\mathsf{a}) = a$; $I_1(\mathsf{b}) = b$; $I_1(\mathsf{c}) = c$
$I_1(\mathsf{male}) = \{a\}$
$I_1(\mathsf{female}) = \{b, c\}$
$I_1(\mathsf{likes}) = \{\langle a,c\rangle, \langle b,c\rangle, \langle c,c\rangle\}$
$I_1(\mathsf{sibling}) = \{\langle a,b\rangle, \langle b,a\rangle, \langle a,c\rangle, \langle c,a\rangle, \langle b,c\rangle, \langle c,b\rangle\}$

(ii)



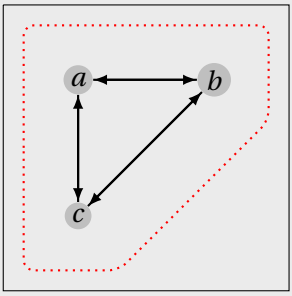
$M_2 = \langle D, I_2\rangle$
$D = \{a, b, c\}$
$I_2(\mathsf{a}) = a$; $I_2(b) = b$; $I_2(c) = c$
$I_2(\mathsf{male}) = \{a, c\}$
$I_2(\mathsf{female}) = \{a, b\}$
$I_2(\mathsf{likes}) = \{\langle a,b\rangle, \langle a,c\rangle, \langle b,a\rangle, \langle c,c\rangle\}$
$I_2(\mathsf{sibling}) = \{\langle a,c\rangle, \langle c,a\rangle\}$

(iii)



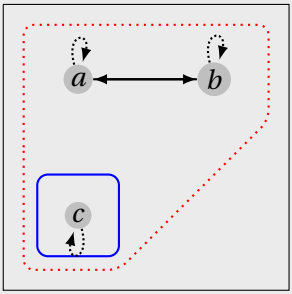
$M_3 = \langle D, I_3\rangle$
$D = \{a, b, c\}$
$I_3(\mathsf{a}) = a$; $I_3(b) = b$; $I_3(c) = c$
$I_3(\mathsf{male}) = \{c\}$
$I_3(\mathsf{female}) = \{a\}$
$I_3(\mathsf{likes}) = \{\langle a,a\rangle, \langle a,b\rangle, \langle c,c\rangle\}$
$I_3(\mathsf{sibling}) = \{\langle a,c\rangle, \langle c,a\rangle\}$

(iv)



$M_4 = \langle D, I_4 \rangle$
$D = \{a, b, c\}$
$I_4(\mathsf{a}) = a$; $I_4(\mathsf{b}) = b$; $I_4(\mathsf{c}) = c$
$I_4(\mathsf{male}) = \varnothing$
$I_4(\mathsf{female}) = \{a, b, c\}$
$I_4(\mathsf{likes}) = \varnothing$
$I_4(\mathsf{sibling}) = \{\langle a, b\rangle, \langle b, a\rangle, \langle a, c\rangle, \langle c, a\rangle, \langle b, c\rangle, \langle c, b\rangle\}$

(v)



$M_5 = \langle D, I_5 \rangle$
$D = \{a, b, c\}$
$I_5(\mathsf{a}) = a$; $I_5(\mathsf{b}) = b$; $I_5(\mathsf{c}) = c$
$I_5(\mathsf{male}) = \{c\}$
$I_5(\mathsf{female}) = \{a, b, c\}$
$I_5(\mathsf{likes}) = \{\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle\}$
$I_5(\mathsf{sibling}) = \{\langle a, b\rangle, \langle b, a\rangle\}$

4.3 Summary

4.3.1 Syntax of $\mathrm{L}_{\mathrm{Pred}}$

The syntax of $\mathrm{L}_{\mathrm{Pred}}$ is defined as follows.

1. **Basic Expressions**

   - **Individual constants:** a, b, c, d, e, f
   - **Individual variables:** x_n, y_n, and z_n for every natural number n;
     x, y, and z;
     and x', x'', etc.
   - **Predicate symbols**

  - Unary: happy, ...
  - Binary: loves, ...

2. **Terms**

   - Every individual constant is a term.
   - Every individual variable is a term.

3. **Atomic formulas**

   - **Predication**
     If π is a predicate of arity n and $\alpha_1, \ldots \alpha_n$ is a sequence of terms, then $\pi(\alpha_1, \ldots \alpha_n)$ is an atomic formula.

     Special cases:
     - If π is a unary predicate and α is a term, then $\pi(\alpha)$ is a formula.
     - If π is a binary predicate and α and β are terms, then $\pi(\alpha, \beta)$ is a formula.
   - **Identity**
     If α and β are terms, then $\alpha = \beta$ is an atomic formula.

4. **Negation**

   - If ϕ is a formula, then $\neg\phi$ is a formula.

5. **Binary connectives**
   If ϕ is a formula and ψ is a formula, then so are:

   - $[\phi \wedge \psi]$ — 'ϕ and ψ'
   - $[\phi \vee \psi]$ — 'ϕ or ψ'
   - $[\phi \rightarrow \psi]$ — 'if ϕ then ψ'
   - $[\phi \leftrightarrow \psi]$ — 'ϕ if and only if ψ'

6. **Quantifiers**
   If u is a variable and ϕ is a formula, then both of the following are formulas:

- $[\forall u.\phi]$ 'for all u: ϕ'
- $[\exists u.\phi]$ 'there exists a u such that ϕ'

We want to avoid unnecessary clutter in our representations, so as mentioned above, we allow brackets to be dropped when it is independently clear what the scope of a quantifier is, and we also allow the outermost brackets of an expression to be dropped. For example, instead of:

$$[\forall x[\mathsf{linguist}(x) \rightarrow [\exists y.\mathsf{admires}(x,y)]]]$$

we can write:

$$\forall x[\mathsf{linguist}(x) \rightarrow \exists y.\mathsf{admires}(x,y)]$$

because it is clear that the scope of the existential quantifier does not extend any farther to the right than it does. Furthermore, when reading a formula, you may assume that the scope of a binder (e.g. $\forall x$ or $\exists x$) extends as far to the right as possible. So, for example, $\forall x[P(x) \wedge Q(x)]$ can be rewritten as $\forall x.P(x) \wedge Q(x)$, interpreted in such a way that the universal quantifier takes scope over the conjunction, rather than as the conjunction of $\forall x.P(x)$ and $Q(x)$. (As a heuristic, you may think of the dot as a "wall" that forms the left edge of a constituent, which continues until you find an unbalanced right bracket or the end of the expression.) However, we will typically retain brackets around conjunctions, disjunctions, and implications.

We retain all of the abbreviatory conventions from above in order to avoid unnecessary clutter in our formulas. Furthermore, we can drop the dot between two quantificational binders in a row. Thus instead of:

$$\forall x.\exists y.\mathsf{admires}(x,y)$$

we can write:

$$\forall x\exists y.\mathsf{admires}(x,y)$$

This convention is specific to our textbook, and there is no single standard in the field. In the Lambda Calculator, on its default setting, dots are *always* optional.

4.3.2 Semantics of L_{Pred}

Now for the semantics of L_{Pred}. The semantic value of an expression is determined relative to two parameters:

1. a model $M = \langle D, I \rangle$ where D is the set of individuals and I is a function mapping each non-logical constant of the language to an element, subset, or relation over elements in D, depending on the nature of the constant;

2. an assignment function g mapping each individual variable in L_{Pred} to some element in D.

For any given model M and assignment function g, the denotation of a given expression α relative to M and g, written $[\![\alpha]\!]^{M,g}$, is defined as follows:

1. **Non-logical constants**

   - If α is a non-logical constant, then $[\![\alpha]\!]^{M,g} = I(\alpha)$.
   - If α is a variable, then $[\![\alpha]\!]^{M,g} = g(\alpha)$.

2. **Atomic formulas**

   - **Predication**
     If π is a predicate of arity n and $\alpha_1, ..., \alpha_n$ is a sequence of terms, then: $[\![\pi(\alpha_1, ...\alpha_n)]\!]^{M,g} = \mathsf{T}$ if $\langle [\![\alpha_1]\!]^{M,g}, ..., [\![\alpha_n]\!]^{M} \rangle \in [\![\pi]\!]^{M,g}$, and F otherwise.[4]
   - **Identity**
     If α and β are terms, then

[4] Special cases:

– When π is a predicate of arity 1, then:

$$[\![\pi(\alpha)]\!] = \mathsf{T} \text{ if } [\![\alpha]\!]^{M,g} \in [\![\pi]\!]^{M,g} \text{ and } \mathsf{F} \text{ otherwise.}$$

– When π is a predicate of arity 2, then:

$$[\![\pi(\alpha, \beta)]\!] = \mathsf{T} \text{ if } \langle [\![\alpha]\!]^{M,g}, [\![\beta]\!]^{M,g} \rangle \in [\![\pi]\!]^{M,g} \text{ and } \mathsf{F} \text{ otherwise.}$$

$$[\![\alpha = \beta]\!]^{M,g} = \mathsf{T} \text{ if } [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g},$$

and F otherwise.

3. **Negation**
   - $[\![\neg\phi]\!]^{M,g} = \mathsf{T}$ if $[\![\phi]\!]^{M,g} = \mathsf{F}$, and F otherwise.
4. **Binary Connectives**
   - $[\![\phi \wedge \psi]\!]^{M,g} = \mathsf{T}$ if $[\![\phi]\!]^{M,g} = \mathsf{T}$ and $[\![\psi]\!]^{M,g} = \mathsf{T}$, and F otherwise.
   - $[\![\phi \vee \psi]\!]^{M,g} = \mathsf{T}$ if $[\![\phi]\!]^{M,g} = \mathsf{T}$ or $[\![\psi]\!]^{M,g} = \mathsf{T}$, and F otherwise.
   - (Semantic rules for $\rightarrow$ and $\leftrightarrow$ were left as exercises.)
5. **Quantification**
   - $[\![\forall v\,.\,\phi]\!]^{M,g} = \mathsf{T}$ if for all individuals $k \in D$:

     $$[\![\phi]\!]^{M,g[v \mapsto k]} = \mathsf{T}$$

     and F otherwise.
   - $[\![\exists v\,.\,\phi]\!]^{M,g} = \mathsf{T}$ if there is an individual $k \in D$ such that:

     $$[\![\phi]\!]^{M,g[v \mapsto k]} = \mathsf{T}$$

     and F otherwise.

4.3.3 Entailment and validity in $\mathrm{L}_{\mathrm{Pred}}$

Because formulas of $\mathrm{L}_{\mathrm{Pred}}$ may contain free variables, their denotations are computed relative to an assignment function as well as a model. We therefore say that ϕ is TRUE IN A MODEL M if and only if $[\![\phi]\!]^{M,g} = \mathsf{T}$ for every assignment g, and false in M if and only if $[\![\phi]\!]^{M,g} = \mathsf{F}$ for every assignment g. For a formula with no free variables, the choice of g makes no difference, so every such formula is either true in M or false in M.

With this in place, the notions of Chapter 1 carry over from $\mathrm{L}_{\mathrm{Prop}}$, with models playing the role that interpretations played there:

- $\phi_1, \ldots, \phi_n \vDash \psi$ if and only if ψ is true in every model in which all of $\phi_1, \ldots, \phi_n$ are true.
- An argument is VALID if and only if its premises entail its conclusion.
- $\phi \equiv \psi$ if and only if $[\![\phi]\!]^{M,g} = [\![\psi]\!]^{M,g}$ for every model M and every assignment g.
- ϕ is SATISFIABLE if and only if it is true in at least one model, and VALID AS A FORMULA if and only if it is true in every model.

5 Typed lambda calculus

5.1 Introduction

As you may recall from the introduction, this book develops a system that assigns truth conditions to sentences in a compositional manner, with the semantic values of larger expressions built up from those of the parts of these expressions. Following Frege, we adopt the idea that semantic composition involves a kind of saturation that can be modelled using functions. Suppose you have a syntactic phrase consisting of two sub-phrases, such as a sentence made up of a subject and a verb phrase, or a verb phrase made up of a transitive verb and its object. In order for the semantic values of the sub-phrases to combine via saturation, one of them must denote a function and the other must denote a potential argument to that function. Currently, we have only a very limited set of tools for describing functions. In this chapter, we will expand our range of tools. Doing so will enable us to model semantic composition in an elegant and general way.

Natural languages have a powerful device for building new predicates out of existing material: the RELATIVE CLAUSE. Consider the two phrases:

(1) a. *person who Blake loves*
    b. *person who loves Blake*

Both phrases denote a property—a set of individuals—and both are assembled from the same two words, *loves* and *Blake*. Yet they

denote *different* properties: (1)a holds of those who are loved by Blake, while (1)b holds of those who love Blake. The difference is which argument position is left open as a GAP: in (1)a the gap is in the theme position (the slot normally filled by the direct object), and in (1)b it is in the agent position. By leaving different positions open, the relative clause constructs new, non-lexical predicates from the same basic material.

A formal theory of meaning must be able to represent these two properties as distinct objects. The predicate logic we have used so far does not give us a direct way to do this. We can translate the sentence *Blake loves Alex* as $\mathsf{loves}(\mathsf{b},\mathsf{a})$, and we can name primitive properties with predicate constants like tall. But $\mathrm{L}_{\mathrm{Pred}}$ provides no device for *constructing* new property expressions from existing ones—no formal counterpart of the relative clause.

The language of the SIMPLY TYPED LAMBDA CALCULUS, developed by the logician Alonzo Church, provides exactly this counterpart. Using an ABSTRACTION OPERATOR, written as the Greek letter λ ('lambda'), we introduce a variable to mark the gap position and bind it, yielding a LAMBDA EXPRESSION. The two phrases in (1) can now be translated:

(2) a. *person who Blake loves* $\rightsquigarrow \lambda x.\,\mathsf{loves}(\mathsf{b},x)$
b. *person who loves Blake* $\rightsquigarrow \lambda x.\,\mathsf{loves}(x,\mathsf{b})$

The expression $\lambda x.\,\mathsf{loves}(\mathsf{b},x)$, read 'lambda x dot loves b x', denotes a function from individuals to truth values that maps any individual x to true if and only if Blake loves x. The expression $\lambda x.\,\mathsf{loves}(x,\mathsf{b})$ maps x to true if and only if x loves Blake. These are two distinct functions—the formal counterparts of the two distinct predicates in (1)—produced from the same predicate constant loves and individual constant b simply by varying which argument position the variable fills. This device is known as LAMBDA ABSTRACTION. (These translations set aside the contribution of the noun *person*.)

The same device applies directly to verb phrases. As Frege put

it, a VP like *loves Blake* expresses something unsaturated: a function waiting for an argument to fill the open slot. The VP leaves the agent position open and receives the same translation as (2)b:

(3) *loves Blake* $\rightsquigarrow \lambda x.\mathsf{loves}(x,\mathsf{b})$

In Chapter 2, we assumed that verb phrases (as well as nouns) denote sets of individuals; replacing sets by their characteristic functions, we are making the same assumption here.

The abstraction operator is also useful in the analysis of natural language expressions like *Everything*. A sentence containing *everything* is always translated as something of the following form:

$$\forall x.\_\_\_\_(x)$$

where ____ is a placeholder for some predicate. For instance, *Everything is temporary* could be expressed:

(4) $\forall x.\mathsf{temporary}(x)$

while *Everything is permanent* would be expressed:

(5) $\forall x.\mathsf{permanent}(x)$

What is constant across these uses is the universal quantification; only the predicate varies. We can capture this if we can abstract over the predicate. Suppose that P is a variable over predicates; then we can abstract over that position using the following expression:

(6) $\lambda P.\forall x.P(x)$

This lambda expression (read 'lambda P (dot) for all x, P of x') denotes a function that expects a predicate, and returns a truth value that depends on the input predicate. More specifically, it denotes a function from a predicate P to a truth value: true if everything satisfies P, and false otherwise. An expression that applies to a function in this way is called a HIGHER-ORDER EXPRESSION, and

the function it denotes a HIGHER-ORDER FUNCTION. So *everything* is a higher-order expression, and so is the determiner *every*, whose translation we come to below.

Now, so far we have not had any variables over predicates. In first-order logic, which we have been using so far, variables only range over individuals. From here on in, we will be using a HIGHER-ORDER LOGIC. This means we can have variables ranging over predicates, which can then be abstracted over and quantified over. It also means that expressions other than terms can serve as arguments to other expressions. So the logic in this chapter is different from L_{Pred} in two respects: it contains lambda abstraction, and it is a higher-order logic.

In this chapter, we will define the syntax and semantics of a language that includes this lambda operator. We will name the language L_λ, after its most important symbol.

5.2 Lambda abstraction, types, and currying

Our languages L_{Prop} and L_{Pred} had a rather limited set of syntactic categories: terms, predicates of various arities, and formulas. In the language L_λ that we present next, we will have a much richer set of syntactic categories, called TYPES. Strictly speaking, a type is a syntactic category for an expression of the logic, but a type also represents the kind of denotation an expression has, and puts constraints on which other expressions (if any) the expression can combine with.

The set of types is *recursively specified*, so they can be of arbitrary complexity and depth, but there are strict rules as to what counts as a type and what doesn't. We will start with two BASIC TYPES:

(7) e

(the type of entities) for individual-denoting expressions (corresponding to TERMS in Predicate Logic), and

(8) t

(the type of truth values) for formulas.

From now on, we will use the term EXPRESSION for any well-formed string of any type, and the term FORMULA for any expression of type t. We will assign types in such a way that everything that was a formula in propositional or first-order predicate logic will continue to be a formula.

From these types we will build up FUNCTION TYPES such as:

(9) $\langle e, t \rangle$

for expressions denoting functions from individuals to truth values. These expressions will include one-place predicates over individuals like temporary and certain lambda expressions like $\lambda x\,.\,\text{loves}(x, \text{b})$. The set of types is defined recursively as follows, where σ 'sigma' and τ 'tau' are not themselves types but rather are meta-variables that stand for arbitrary types:

- e is a type.
- t is a type.
- If σ is a type and τ is a type, then the ordered pair $\langle \sigma, \tau \rangle$ is a type.
- Nothing else is a type.

For example, $\langle e, t \rangle$ is a type, since both e (our σ) and t (our τ) are types. Note that σ and τ could in principle be instantiated by the same actual type; for example, $\langle e, e \rangle$ is a type, since σ and τ don't have to be distinct. Also, since $\langle e, t \rangle$ is a type, and e is a type of course, it follows that $\langle e, \langle e, t \rangle \rangle$ is a type. And so on. The set of types is infinite.

These types are *syntactic categories* of expressions of our logical language. In any given model, these expressions denote various kinds of objects, and in this way types are indirectly associated with the objects that these expressions denote. A model as-

sociates each type with a different DOMAIN, the set of possible denotations for expressions of that type. For any type τ, we use D_τ to signify the set of possible denotations for an expression of type τ. An expression of type e denotes an individual; D_e is the set of individuals. So we say indirectly that e is the type of individuals. An expression of type t is a formula, so its denotation must be either T or F; $D_t = \{\mathsf{T}, \mathsf{F}\}$. An expression of type $\langle e, t\rangle$ denotes a function from individuals to truth values. $D_{\langle e,t\rangle}$ is the set of functions with domain D_e and codomain D_t; that is, functions that take as input an individual, and give a truth value as output. An expression of type $\langle e, \langle e, t\rangle\rangle$ denotes a function which takes an individual as its input and returns a function from individuals to truth values. An expression of type $\langle\langle e, t\rangle, e\rangle$ denotes a function which takes a function from individuals to truth values as its input and returns an individual. And so forth.

Although the set of types is infinite, there are limits: Not everything is a type. For example, $\langle e\rangle$ is *not* a type according to this system (though some authors write $\langle e\rangle$ for e); according to our definition, angle brackets are only introduced for *function types*, which are ordered pairs of types.

The system used in this book and in the foundational formal semantic work by Montague also lacks types corresponding to sets and binary relations, the sorts of things that the unary and binary predicates of predicate logic denote. In this language, an expression cannot denote a set, because there is no type for that. There is, however, the type $\langle e, t\rangle$, which corresponds to the characteristic function of a set (a function that takes an individual, and returns true or false depending on whether that individual is in the set). From the characteristic function of a set, one can figure out what the members of the set are (it is the characteristic set of that function), so unary predicates can be replaced by expressions of type $\langle e, t\rangle$ with no loss of information.

Similarly, an expression cannot denote a binary relation, as there is no type for that. But we do have the type $\langle e, \langle e, t\rangle\rangle$, which

can encode a binary relation, by using a method known as CURRYING.[1] Currying serves to change a single function taking multiple arguments into multiple functions each taking a single argument. For example, a binary relation R is a set of pairs of individuals. The characteristic function of this set is a function that applies to pairs of individuals and returns a truth value. From this characteristic function, LEFT-TO-RIGHT CURRYING produces a function f such that $[f(x)](y) = \top$ if and only if $\langle x, y\rangle \in R$ (where $[f(x)](y)$ denotes the result of first applying f to x, and then applying $f(x)$ to y). Analogously, right-to-left currying produces a function f such that $[f(x)](y) = \top$ if and only if $\langle y, x\rangle \in R$.

For example, imagine that the 'love' relation over the set

{Alex,Blake,Casey}

is as follows:

(10) $\{\langle \text{Casey}, \text{Blake}\rangle, \langle \text{Casey}, \text{Casey}\rangle\}$

Casey loves Blake, Casey loves Casey, and nobody loves anyone else. This relation can be recoded as a function of type $\langle e, \langle e, t\rangle\rangle$ that would be able to serve as the meaning for the verb *loves* using Currying.

We start with the characteristic function of this relation, call it f, shown below. Applied to any pair of individuals, it returns a truth value: $\top$ if that pair is in the relation, F otherwise.

[1] This procedure is named after the logician Haskell Curry. It is also called 'Schönfinkelization', after the logician Moses Schönfinkel, on whose work Curry built. See Heim & Kratzer (1998) p. 41, fn. 13. Hindley & Seldin (2008, p. 3) write, "Curry always insisted that he got the idea of using [curried functions] from [Schönfinkel 1924 (see Curry & Feys 1958, pp. 8, 10)], but most workers seem to prefer to pronounce 'currying' rather than 'schönfinkeling'. The idea also appeared in 1893 in [Frege 1983, Vol. 1, Section 4]."

$$(11)\qquad f = \begin{bmatrix} \langle \text{Alex}, \text{Alex}\rangle & \to & \mathsf{F} \\ \langle \text{Alex}, \text{Blake}\rangle & \to & \mathsf{F} \\ \langle \text{Alex}, \text{Casey}\rangle & \to & \mathsf{F} \\ \langle \text{Blake}, \text{Alex}\rangle & \to & \mathsf{F} \\ \langle \text{Blake}, \text{Blake}\rangle & \to & \mathsf{F} \\ \langle \text{Blake}, \text{Casey}\rangle & \to & \mathsf{F} \\ \langle \text{Casey}, \text{Alex}\rangle & \to & \mathsf{F} \\ \langle \text{Casey}, \text{Blake}\rangle & \to & \mathsf{T} \\ \langle \text{Casey}, \text{Casey}\rangle & \to & \mathsf{T} \end{bmatrix}$$

Left-to-right currying turns f into the function we call $f_{\to}$:

$$(12)\qquad f_{\to} = \begin{bmatrix} \text{Alex} & \to & \begin{bmatrix} \text{Alex} & \to & \mathsf{F} \\ \text{Blake} & \to & \mathsf{F} \\ \text{Casey} & \to & \mathsf{F} \end{bmatrix} \\ \text{Blake} & \to & \begin{bmatrix} \text{Alex} & \to & \mathsf{F} \\ \text{Blake} & \to & \mathsf{F} \\ \text{Casey} & \to & \mathsf{F} \end{bmatrix} \\ \text{Casey} & \to & \begin{bmatrix} \text{Alex} & \to & \mathsf{F} \\ \text{Blake} & \to & \mathsf{T} \\ \text{Casey} & \to & \mathsf{T} \end{bmatrix} \end{bmatrix}$$

Right-to-left currying turns f into the function we call $f_{\leftarrow}$:

$$(13)\qquad f_{\leftarrow} = \begin{bmatrix} \text{Alex} & \to & \begin{bmatrix} \text{Alex} & \to & \mathsf{F} \\ \text{Blake} & \to & \mathsf{F} \\ \text{Casey} & \to & \mathsf{F} \end{bmatrix} \\ \text{Blake} & \to & \begin{bmatrix} \text{Alex} & \to & \mathsf{F} \\ \text{Blake} & \to & \mathsf{F} \\ \text{Casey} & \to & \mathsf{T} \end{bmatrix} \\ \text{Casey} & \to & \begin{bmatrix} \text{Alex} & \to & \mathsf{F} \\ \text{Blake} & \to & \mathsf{F} \\ \text{Casey} & \to & \mathsf{T} \end{bmatrix} \end{bmatrix}$$

Both $f_{\to}$ and $f_{\leftarrow}$ are of type $\langle e, \langle e, t\rangle\rangle$. Applied to a given individual x, each one returns another function, which in turn maps

individuals y to truth values: $f_{\rightarrow}$ returns T iff x stands in the original relation to y, and $f_{\leftarrow}$ returns T iff y stands in the original relation to x. For example, there is one ordered pair in the relation whose second element is Casey, namely, $\langle$ Casey, Casey $\rangle$ (we look at the second element because the relation is right-to-left curried.) Accordingly, when we apply $f_{\leftarrow}$ to Casey, the result is a function that maps Casey to T and the others to F:

$$f_{\leftarrow}(\text{Casey}) = \begin{bmatrix} \text{Alex} & \rightarrow & \mathsf{F} \\ \text{Blake} & \rightarrow & \mathsf{F} \\ \text{Casey} & \rightarrow & \mathsf{T} \end{bmatrix} \tag{14}$$

Applied to Casey, it gives back the characteristic function of the set of individuals that love Casey. As another example, when $f_{\leftarrow}$ is applied to Alex, it returns a function that maps everything to F because there is no ordered pair in the relation whose second element is Alex. (Nobody loves Alex.) And so on.

As it turns out, right-to-left currying is precisely what we need in order to give a compositional analysis of sentences in natural languages containing transitive verbs. (We use right-to-left currying because of a mismatch: in bottom-up syntactic derivations, the first argument with which a transitive verb merges is its object, and this order will be mirrored in the compositional semantics. But because subjects occur to the left of objects in many languages, it is customary to think of binary relations as relating subjects to objects in that order. That is, the first element in the pair of a binary relation is thought of as the subject, and the second element is thought of as the object. If this was the other way around, we would use left-to-right currying instead.) In the next chapter, we will characterize transitive verbs as denoting such curried relations – functions which, when given an individual, return *another function*. Rather than translating the verb *loves* as the binary predicate loves, we will translate it as a function that applies to its object (say, *Blake*, in *Alex loves Blake*) to return a new function, which then may apply to the subject (say, *Alex*). That way, every

part of the sentence is assigned a denotation, including the verb phrase (*loves Blake*), and the composition proceeds through the successive application of functions.

To translate the verb *loves*, we can use a simple expression of type $\langle e, \langle e, t\rangle\rangle$ like

(15) loves

so that

(16) $\mathsf{loves(b)}$

is an expression of type $\langle e, t\rangle$ which will serve as the translation for the verb phrase *loves Blake*, and

(17) $\mathsf{loves(b)(a)}$

is an expression of type t (that is, a formula) which will serve as the translation for *Alex loves Blake*. Note that in $\mathsf{loves(b)(a)}$, the subexpression $\mathsf{loves(b)}$ forms a unit. We have $\mathsf{loves(b)(a)}$ rather than $\mathsf{loves(a)(b)}$ because the verb combines first with the object *Blake* and then with the subject *Alex*. But as we generally prefer to read the subject before the object, and in order to reduce parenthesis clutter, we will introduce the following notational convention: instead of $\mathsf{loves(b)(a)}$, we will write *as a shorthand*:

(18) $\mathsf{loves(a, b)}$

We will stick to this RELATIONAL STYLE (as opposed to the FUNCTIONAL STYLE) throughout the book as much as possible. Thus instead of the functional style (19a), with two sets of parens, we will represent the denotation of a transitive verb in lambda calculus in the relational style as in (19b), with just one set of parens:

(19) a. $\lambda y.\lambda x.\mathsf{loves}(y)(x)$
b. $\lambda y.\lambda x.\mathsf{loves}(x, y)$

Both of these expressions denote the same thing as loves in L_λ:

a certain function from individuals to functions from individuals to truth values. In L_λ, these three expressions all denote the result of right-to-left currying the binary relation that is denoted in L_{Pred} by the binary predicate loves. Notice what has become of the inventory of basic expressions. In L_{Pred}, individual constants, unary predicate symbols and binary predicate symbols belonged to three separate syntactic categories. In L_λ they do not: a, smokes and loves are all simply non-logical constants, distinguished from one another only by their types — e, $\langle e,t\rangle$ and $\langle e,\langle e,t\rangle\rangle$ respectively. The work that was previously done by syntactic category is now done by the type system.

Using the relational style in (19b) helps bring out visually how many arguments the verb expects to combine with, and is more similar to how verbal denotations are commonly represented following the style of Heim & Kratzer (1998); the denotation of the verb *loves* in the style of that textbook would be represented as '$\lambda y.\lambda x.x$ loves y', with a blend of English and lambda calculus.

5.3 Syntax and semantics

The lambda operator allows us to describe a wide range of functions. For example:

(20) $\lambda x.\mathsf{loves}(\mathsf{b},x)$

denotes the characteristic function of the set of individuals that Blake loves, while

(21) $\lambda x.\mathsf{loves}(x,\mathsf{b})$

denotes the characteristic function of the set of individuals that love Blake. You can think of the λ-operator analogously to predicate notation for building sets. $\lambda x.\mathsf{loves}(\mathsf{b},x)$ denotes the characteristic function of the set $\{x \mid \text{Blake loves } x\}$, that is, of the set of individuals that Blake loves. (As we noted in Chapter 2, it is common not to distinguish between sets and their characteristic

functions. So we will often also say slightly imprecise things like "$\lambda x.\mathsf{loves}(\mathsf{b}, x)$ denotes the set of individuals that Blake loves.")

The lambda expressions in the previous paragraph are of type $\langle e, t\rangle$, because the input is an individual (something in D_e) and the output is a truth value (something in D_t). In general, if ϕ is a formula (type t), and x is a variable of type e, then $\lambda x.\phi$ will be an expression of type $\langle e, t\rangle$. But the input and the output can be any type whatsoever. Here is a lambda expression of type $\langle e, e\rangle$, assuming that archenemyof is a constant of type $\langle e, e\rangle$ denoting a function from an individual to that individual's arch enemy:

(22) $\lambda x.\mathsf{archenemyof}(\mathsf{archenemyof}(x))$

This function takes as input an individual x and returns as output another individual, the arch enemy of x's arch enemy (which might be expected to be equal to x).

The syntax rule that introduces lambda expressions into the language thus allows for any possible type:

Syntax Rule: Lambda abstraction
If α is an expression of type τ and u is a variable of type σ then $[\lambda u.\alpha]$ is an expression of type $\langle \sigma, \tau\rangle$.

(We will often drop the outer square brackets when it does not result in confusion.) In a lambda expression of the form described in this rule, we call σ and τ the INPUT TYPE and OUTPUT TYPE.

The semantics of lambda expressions is defined as follows:

Semantic Rule: Lambda abstraction
If α is an expression of type τ and u a variable of type σ then for any assignment g, $[\![\lambda u.\alpha]\!]^{M,g}$ is that function f from D_σ into D_τ such that for all objects o in D_σ, $f(o) = [\![\alpha]\!]^{M,g[u\mapsto o]}$.

For example, $\lambda x.\mathsf{happy}(x)$ is of the form $\lambda u.\alpha$ where u (i.e., x) is of type e, and α (i.e., $\mathsf{happy}(x)$) is of type t. So it denotes the

function f from D_e to D_t such that for all objects o in D_e, $f(o)$ is equal to $[\![\mathsf{happy}(x)]\!]^{M,g[x\mapsto o]}$. For any object o, $f(o)$ will return T (True) if o is in the set that happy denotes in M, and F (False) if not. So in an intended model, $\lambda x.\mathsf{happy}(x)$ denotes the characteristic function of the set of happy individuals.

In the indirect interpretation theory, the syntactic constituent "loves Blake" is first translated to the expression $\lambda x.\mathsf{loves}(x,\mathsf{b})$. We will symbolize the translation relation with the symbol $\rightsquigarrow$ (pronounced "translates to" or "is translated as"). Then, the $[\![\cdot]\!]^{M,g}$ denotation function maps this lambda expression to an appropriate function; if M is an intended model, this is the characteristic function of the set of individuals who love Blake in M. If that characteristic function is given the individual Alex as an input, the output is a truth value: T if Alex loves Blake in M; F if not.

If this seems overwhelming, stay calm; it may start to sink in after you get some practice with beta reduction, which we turn to next.

5.4 Application and beta reduction

The functions resulting from abstraction behave just like the functions we are already familiar with. We indicate the arguments of a function using parentheses, and enclose the whole application in square brackets. An expression consisting of a function followed by its argument is called an APPLICATION. Bear in mind that 'function' here covers what were previously called predicate and relation symbols: smokes, of type $\langle e,t\rangle$, denotes a function from individuals to truth values, so $\mathsf{smokes}(\mathsf{a})$ is an application just as much as $[\lambda x.\mathsf{happy}(x)](\mathsf{a})$ is. If π is an expression denoting a function, and α is an expression whose type is the input type of π, then $[\pi(\alpha)]$ is an application term denoting the result of applying π to α, and its type is the output type of π. (We will often drop the outer square brackets when it does not result in confusion.) For example, $[\lambda x.\mathsf{happy}(x)](\mathsf{a})$ denotes the result of apply-

ing the function denoted by $[\lambda x.\mathsf{happy}(x)]$ to the semantic value of a. This principle also applies to syntactically complex function-denoting terms formed by lambda abstraction. Thus

(23) $[\lambda x.\mathsf{loves}(x,\mathsf{b})](\mathsf{a})$

denotes the result of applying the function 'loves Blake' to Alex.

Here is the syntax rule that introduces function application terms into the language:

Syntax Rule: Application
For any types σ and τ, if α is an expression of type $\langle\sigma,\tau\rangle$ and β is an expression of type σ then $[\alpha(\beta)]$ is an expression of type τ.

The semantics of function application is defined as follows:

Semantic Rule: Application
If α is an expression of type $\langle\sigma,\tau\rangle$, and β is an expression of type σ, then $[\![\alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$.

We call two expressions EQUIVALENT, written $\alpha \equiv \beta$, if and only if in every model they have the same denotation as each other. The order of the quantifiers matters here: the requirement is not that there be some one denotation which both expressions have in every model, but rather that in each model, whatever the first denotes, the second denotes too. A non-logical constant will typically denote different things in different models, but it is of course equivalent to itself, since within any given model it cannot denote two different things. For example, the expression we have just seen is provably equivalent to the simpler:

(24) $\mathsf{loves}(\mathsf{a},\mathsf{b})$

where the λ-binder, the square brackets, and the variable have been removed, and we have kept just the part after the dot, with

the modification that the argument of the function is substituted for all instances of the variable. This kind of simplification is known as BETA REDUCTION (other names include BETA CONVERSION and LAMBDA CONVERSION).

Using beta reduction, an expression of the form:

$$[\lambda x. \ldots\ x\ \ldots](\alpha)$$

can be simplified to

$$\ldots\ \alpha\ \ldots$$

The following pairs of expressions are equivalent; in each case, the second is the beta-reduced version of the first.

(25) a. $[\lambda x.\mathsf{smiled}(x)](\mathsf{a})$
b. $\mathsf{smiled}(\mathsf{a})$

(26) a. $[\lambda x.[\mathsf{smiled}(x) \wedge \mathsf{happy}(x)]](\mathsf{a})$
b. $[\mathsf{smiled}(\mathsf{a}) \wedge \mathsf{happy}(\mathsf{a})]$

(27) a. $[\lambda x.[\mathsf{smiled}(x) \wedge \mathsf{happy}(y)]](\mathsf{a})$
b. $[\mathsf{smiled}(\mathsf{a}) \wedge \mathsf{happy}(y)]$

In a lambda expression of the form $\lambda x.\phi$, the ϕ part (the scope of the lambda expression) describes the value of the function given an argument, so it can be called the VALUE DESCRIPTION (or BODY). For example, the value description in the expression

(28) $\lambda x.\mathsf{loves}(x,\mathsf{b})$

is

(29) $\mathsf{loves}(x,\mathsf{b})$.

Exercise 1. Identify the value description in the following lambda expressions:

1. $\lambda x.\mathsf{happy}(x)$

2. $\lambda x.x$
3. $\lambda y.\lambda x.[\mathsf{loves}(x,y) \vee \mathsf{loves}(y,x)]$
4. $\lambda z.\lambda y.\lambda x.\mathsf{between}(x,y,z)$

In general, the result of applying a function described by a lambda expression to an argument can itself be described by a new expression, obtained by *taking the value description and replacing all free occurrences of the lambda-bound variable with the argument*. Producing that expression is precisely the beta reduction introduced above. By 'free occurrences', we mean occurrences that are not bound by another variable binder (a lambda operator or a quantifier). The official definition of beta-reduction is as follows. Here we write x for a variable of any type, α for an expression of the type of x, ϕ for an expression of any type, and $\phi[x \coloneqq \alpha]$ for the result of replacing with α all free occurrences of x in ϕ:

Beta reduction: $[\lambda x.\phi](\alpha)$ can be reduced to $\phi[x \coloneqq \alpha]$ provided that α does not contain any free variables that occur bound in ϕ.

If another variable binder is present in the value description and binds the very same variable that is bound by the lambda operator in question, then occurrences of the variable that are in the scope of that other variable binder are no longer bound by the lambda operator. So the following two formulas are equivalent:

(30) a. $[\lambda x.[\mathsf{smiled}(x) \wedge \exists x.\mathsf{happy}(x)]](\mathsf{a})$
b. $[\mathsf{smiled}(\mathsf{a}) \wedge \exists x.\mathsf{happy}(x)]$

The occurrence of the variable x inside the scope of the existential quantifier is bound by that quantifier and not by the lambda operator, so replacing it with a would not result in an equivalent expression.

To avoid confusion, as a matter of practice, it is best to avoid letting the same variable be bound by more than one binder. But if you find yourself in such a situation, you can remedy it using the rule of ALPHA CONVERSION. This is a re-lettering rule: it allows one to replace bound variables by other ones under certain conditions without a change in denotation. For instance, $\forall x.P(x)$ is equivalent to $\forall y.P(y)$, where we have 're-lettered' all occurrences of x as y. The same holds for lambda-bound variables as well: $\lambda x.P(x)$ is equivalent to $\lambda y.P(y)$. Alpha conversion also allows us to convert

(31) $\lambda x.\mathsf{loves}(x,y)$

into

(32) $\lambda z.\mathsf{loves}(z,y)$

Here we replaced x with z, which we could do because z did not already occur free in the value description in (31). We could not have picked y, as that would have produced a VARIABLE COLLISION, also known as an ACCIDENTAL CAPTURE. If you replace the lambda-bound variable with one that already occurs free in the body of the lambda expression (such as y in this example), the lambda operator will come to bind that variable occurrence whereas it did not before, so that would change the meaning. But for any variables u and v, as long as v does not occur free in ϕ, $\lambda u.\phi$ can be written equivalently as $\lambda v.\phi'$, where ϕ' is a version of ϕ with all free instances of u replaced by v.

In the context of beta-reduction, alpha conversion can be especially useful when the argument contains a free variable, or *is* a variable itself. For example, consider:

(33) $[\lambda y.\lambda x.\mathsf{loves}(x,y)](x)$

If we just substitute x in for y, we get:

(34) $\lambda x.\mathsf{loves}(x,x)$

which is not equivalent to (33). The rule of beta-reduction does not allow this substitution, because it contains the proviso, "provided that α [the argument] does not contain any free variables that occur bound in ϕ [the value description]." In this case, the argument (here, x) contains a free variable that occurs in the value description (here, $\lambda x.\mathsf{loves}(x,y)$). And indeed, (34) is not equivalent to the original expression (33). If loves denotes the relation of loving someone, then the expression (34) denotes the set of self-lovers, while the expression (33) denotes the set of those who love whomever x picks out. Through overly enthusiastic substitution of y for x, the variable y accidentally became bound by the inner lambda operator. But the inner lambda expression could have involved any variable. It didn't have to be x. For example, it could have been z. Using alpha conversion, we reletter x as z in (33) and get the following:

(35) $[\lambda y.\lambda z.\mathsf{loves}(z,y)](x)$

This expression has exactly the same denotation as (33). If we substitute x for y by performing beta reduction on (35), then we get the right result: an expression that has the same denotation as (33) (and (35)), but that cannot be simplified any further.

(36) $\lambda z.\mathsf{loves}(z,x)$

Whereas our former attempt in (34) denotes the set of individuals who love themselves, this denotes the set of individuals who love whomever x picks out.

When doing beta reduction on arguments that contain free variables which also occur in the value description, as in (33), we recommend first using alpha conversion to 're-letter' the bound variable as in (35), before carrying out beta reduction as usual. In the Lambda Calculator, this re-lettering procedure is enforced as a matter of practice.[2]

[2] A third rule, ETA REDUCTION, allows us to rewrite a lambda expression of the

5.5 Well-typed expressions and type checking

Not every sequence of symbols is a meaningful expression in the typed lambda calculus. For an expression to be meaningful, or WELL-TYPED, it must be possible to assign a type to it based on the types of its constituent parts. The process of assigning a type to an expression is called TYPE CHECKING or TYPE DERIVATION.

The assignment of types is COMPOSITIONAL: the type of a complex expression is determined by the types of its sub-expressions. This means we can determine the type of any expression by starting with the types of its simplest components (variables and constants), which are given, and then using a few simple rules to determine the types of larger expressions built from these components.

As we have seen, the two syntax rules for building complex ex-

shape $\lambda x.[\phi(x)]$ as just ϕ, and vice versa. For example, this rule ensures that $\lambda x.\mathsf{smiled}(x)$ and $\lambda y.\mathsf{smiled}(y)$ are equivalent to each other and to smiled. Two other examples: $\lambda y.[\lambda x.\mathsf{loves}(y)(x)]$ is equivalent to $\lambda y.\mathsf{loves}(y)$, which in turn is equivalent to loves. This can be another handy way of simplifying representations. Like the other rules, it comes with a proviso regarding free variables: ϕ can be any expression except it must not contain any free occurrences of x. For example, $\lambda x.\mathsf{loves}(x)(x)$ (which denotes the set of self-lovers) cannot be eta-reduced to $\mathsf{loves}(x)$, which denotes the set of x-lovers rather than self-lovers. If it is not clear why $\mathsf{loves}(x)$ has this denotation, it might help to see that it can be obtained via eta reduction from $\lambda y.\mathsf{loves}(x)(y)$, which also denotes the set of x-lovers. (Remember that curried predicates that translate to transitive verbs take their object before their subject.)

Eta reduction is needed in order to derive equivalences that one would not have been able to derive with just alpha and beta reduction. For example, the expressions $\lambda x.[(\lambda P.P)(\mathsf{smiled})](x)$ and smiled have the same denotation in every model, but we cannot reduce the first to the second without using eta-reduction.

One can show that these three rules never change the denotation or the type of a lambda expression and that they always give the same result no matter in which order they are applied to a complex lambda expression. And in the simply typed lambda calculus, one can show that these rules will always terminate, i.e. no matter how complex the initial expression, it is always possible to come to a point where none of these rules can be applied.

pressions involving lambdas are lambda abstraction and function application. The syntax rule for lambda abstraction tells us that if you have a variable v of type σ and an expression α of type τ (which typically contains v), then the lambda abstraction $\lambda v_\sigma . \alpha$ is a well-typed expression of type $\langle \sigma, \tau \rangle$. The syntax rule for application tells us that if you have an expression α of type $\langle \sigma, \tau \rangle$ and an expression β of type σ, then the application of α to β, written as $\alpha(\beta)$, is a well-typed expression of type τ.

Since we are building our lambda terms on top of predicate logic, which in turn builds on propositional logic, the syntax rules from previous chapters continue to be in force, but we now interpret them as specifying and constraining types for expressions. For example, the syntax rule for conjunction says that if you have two expressions α and β, each of type t, then conjunction $[\alpha \wedge \beta]$ is a well-typed expression of type t. The old syntax rule for predication is dropped as it is subsumed by the new and more general syntax rule for application.

This compositional approach allows us to systematically check whether any given lambda expression is well-typed and, if it is, to determine its type. The process can be formalized using a set of inference rules, but for our purposes, an informal, example-driven approach will be sufficient. The following examples illustrate this process in detail.

5.6 Type checking in practice: Examples

Let's walk through some examples to see how type checking works in practice. We will represent the process of type checking using a tree-like structure, where the leaves are the basic expressions (constants or variables) and the nodes are complex expressions formed by function application or lambda abstraction. In order to check the type of a complex expression, one needs to know the types of its smallest subexpressions. In the examples that follow, we will use subscripts to indicate their types. In practice, one of-

ten leaves out these subscripts and implicitly assumes that certain typing conventions are in place.

Consider the expression in (37), which could represent the meaning of an intransitive verb phrase like *snores*.

(37) $\lambda x_e . \mathsf{snores}_{\langle e,t \rangle}(x)$

To check the type of this expression, we start from its innermost parts and work our way out.

1. The constant snores has the type $\langle e, t \rangle$.
2. The variable x has the type e.
3. The sub-expression $\mathsf{snores}(x)$ is a function application. The function, snores, has type $\langle e, t \rangle$, so its input type is e. Since this input type is also the type of the argument x, the application as a whole is well-typed, and its type is the output type of the function, which is t.
4. The entire expression is a lambda abstraction over the variable x (type e) and the body $\mathsf{snores}(x)$ (type t). The type of the entire expression is therefore $\langle e, t \rangle$.

The expression is well-typed and has the type $\langle e, t \rangle$.

Now consider a slightly more complex expression, representing a transitive verb like *loves*:

(38) $\lambda y_e . \lambda x_e . \mathsf{loves}_{\langle e,\langle e,t \rangle \rangle}(y)(x)$

The type checking process is as follows:

1. The constant loves has the type $\langle e, \langle e, t \rangle \rangle$.
2. The variable y has the type e.
3. The sub-expression $\mathsf{loves}(y)$ is a well-typed application, since loves (type $\langle e, \langle e, t \rangle \rangle$) takes an argument of type e (y). The result is an expression of type $\langle e, t \rangle$.

4. The variable x has the type e.

5. The sub-expression $\mathsf{loves}(y)(x)$ is also a well-typed application. The function, $\mathsf{loves}(y)$, has type $\langle e,t\rangle$ and its argument, x, has type e. The resulting type is t.

6. The expression $\lambda x_e.\mathsf{loves}(y)(x)$ is a lambda abstraction over x (type e) and its body has the type t. The type of $\lambda x_e.\mathsf{loves}(y)(x)$ is therefore $\langle e,t\rangle$.

7. The entire expression is a lambda abstraction over y (type e) and its body has the type $\langle e,t\rangle$. The final type is thus $\langle e,\langle e,t\rangle\rangle$.

Now let's look at an example of an ill-typed expression. Suppose we try to apply the predicate snores to the predicate man.

(39) $\mathsf{snores}_{\langle e,t\rangle}(\mathsf{man}_{\langle e,t\rangle})$

Here, snores is a function of type $\langle e,t\rangle$, which means it expects an argument of type e. However, the argument provided is man, which is of type $\langle e,t\rangle$. Because the type of the argument does not match the input type of the function, this expression is ILL-TYPED and not a meaningful expression in our system.

Another example of an ill-typed expression results from applying a function to too many arguments. Consider the following:

(40) $\lambda x_e.\mathsf{snores}_{\langle e,t\rangle}(x)(\mathsf{alex}_e)$

Let's try to check its type:

1. As before, $\mathsf{snores}(x)$ is a well-typed expression of type t.

2. The expression then attempts to apply this result (of type t) to the constant alex (of type e).

3. However, an expression of type t (a truth value) is not a function, so it cannot be applied to an argument.

This expression is therefore ILL-TYPED.

Finally, let's consider a higher-order function, such as the one representing the meaning of the quantifier *every*:

(41) $\lambda P_{\langle e,t\rangle}.\lambda Q_{\langle e,t\rangle}.\forall x_e.[P(x) \rightarrow Q(x)]$

1. The variables P and Q are of type $\langle e,t\rangle$. The variable x is of type e.

2. The sub-expressions $P(x)$ and $Q(x)$ are both well-typed and have type t.

3. The expression $[P(x) \rightarrow Q(x)]$ is a formula of propositional logic, which evaluates to a truth value. So it is of type t.

4. The expression $\forall x_e.[P(x) \rightarrow Q(x)]$ is a quantified formula, which is also of type t.

5. The abstraction $\lambda Q_{\langle e,t\rangle}.\forall x_e.[P(x) \rightarrow Q(x)]$ takes a function of type $\langle e,t\rangle$ (namely, Q) and returns an expression of type t. Its type is therefore $\langle\langle e,t\rangle,t\rangle$.

6. The entire expression is an abstraction over P (type $\langle e,t\rangle$) and a body of type $\langle\langle e,t\rangle,t\rangle$. The final type is $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$.

This example shows that the same type-checking principles apply to higher-order functions, allowing us to build complex meanings in a systematic and well-defined way.

5.7 Some linguistic applications

Our new and improved representation language, with its capacity for abstraction and its infinitely many types, can express a wide range of potential denotations for natural language expressions. Take for example the prefix *non-*, as in *non-smoker*. A non-smoker is someone who is not in the set of smokers. If *smoker* denotes the set of people who smoke and translates to this:

(42) $\lambda x.\mathsf{smokes}(x)$

Then *non-smoker* should denote the set of people who don't smoke, and it should translate to this:

(43) $\lambda x.\neg\mathsf{smokes}(x)$

On this analysis, a *non-*P is a member of the set denoted by $\lambda x.\neg P(x)$. So the denotation of *non-* can be thought of as a function that takes as its argument a predicate (say, P) and then returns a new predicate which holds of an individual iff the individual does not satisfy the input predicate P:

(44) $\lambda P.[\lambda x.\neg P(x)]$

If we apply this function to $\lambda x.\mathsf{smokes}(x)$, the result is equivalent to $\lambda x.\neg\mathsf{smokes}(x)$. This correctly captures the fact that a *non-smoker* doesn't smoke. As the prefix *non-* applies to a predicate, rather than an individual, it can be said to denote a higher-order function, that is, a function that applies to other functions. By the same token, *non-* is a higher-order expression.

In the beginning of this chapter, we motivated the use of lambda calculus on the basis of its ability to capture the idea of a template with a slot to be filled, but its ability to represent higher-order functions is another important virtue of this formalism as a way of representing natural language. For example, in Chapter 4, we mentioned that the following sentences can be expressed as a single formula in higher-order logic but not in first-order logic:

(45) a. Napoleon had all the properties of a good general.
b. No two distinct objects have the same properties.

Here are two translations of these sentences into higher-order logic:

(46) $\forall P.[[\exists x.\mathsf{good}(x)\wedge\mathsf{general}(x)\wedge P(x)]\rightarrow P(\mathsf{napoleon})]$

(47) $\neg[\exists x\exists y.[\neg(x=y)\wedge[\forall P.P(x)\leftrightarrow P(y)]]]$

In these formulas, P is not a predicate but a variable over predicates (otherwise, it could not be bound by the universal quantifier). The type of this variable is $\langle e, t \rangle$. In the first-order logic we have encountered in Chapter 4, all variables range over individuals; in other words, the types of all first-order variables are e.

Our lambda calculus also provides the right tools for the compositional treatment of determiners like *every*. Note that the truth conditions of *Every cellist smokes* are expressible in first-order logic: $\forall x.[\mathsf{cellist}(x) \rightarrow \mathsf{smokes}(x)]$. What requires higher-order logic is not the truth conditions themselves, but the *compositional treatment*: if *every* is to combine with its arguments (*cellist* and *smokes*) via Function Application, those arguments must be of type $\langle e, t \rangle$, and so the lexical entry for *every* must bind variables of type $\langle e, t \rangle$ — something first-order logic does not permit. In contrast, certain determiners like *most* require higher-order resources even for their truth conditions: 'Most cellists smoke' (more than half of all cellists smoke) cannot be expressed in first-order logic at all, though it can be expressed in the higher-order logic we are developing here.[3]

Recall that intuitively, *every* expresses the subset relation between two sets. To say *Every cellist smokes* is to say that the set of cellists is a subset of the set of individuals that smoke. Let P and Q be variables ranging over the characteristic functions of sets (type $\langle e, t \rangle$). The denotation of *every* can be represented like this:

(48) $\lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x)$

This expression denotes a function that takes a predicate (call it P), and returns a function that takes another predicate (call it Q), and returns $\top$ (True) if and only if every P is a Q.

The denotation of *every cellist* would be the result of applying this function to the denotation of *cellist*. This means that *cellist* must denote a function from individuals to truth values. This sug-

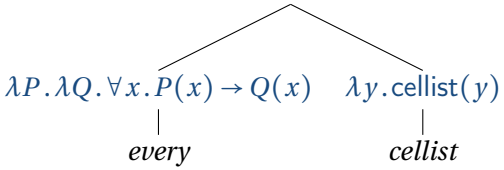
[3]The cardinality comparison required by *most* is not first-order expressible (Barwise & Cooper, 1981).

gests that *cellist* is translated as follows:

(49) $\lambda y.\mathsf{cellist}(y)$

Then *every cellist* will be translated as:

(50) $[\lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x)](\lambda y.\mathsf{cellist}(y))$



The translation at the top can be simplified via two beta reduction. In a first step, we remove λP and correspondingly replace P in the value description by $\lambda y.\mathsf{cellist}(y)$. This gives us:

(51) $\lambda Q.\forall x.[\lambda y.\mathsf{cellist}(y)](x) \rightarrow Q(x)$

In a second step, we apply $\lambda y.\mathsf{cellist}(y)$ to x and get $\mathsf{cellist}(x)$. This happens within the bigger expression, so we end up with:

(52) $\lambda Q.\forall x.\mathsf{cellist}(x) \rightarrow Q(x)$

Thus the denotation of *every*, applied to the denotation of *cellist*, is a function that is still hungry for another unary predicate. Feeding it $\lambda z.\mathsf{smokes}(z)$ produces a formula that denotes a truth value:

(53) $[\lambda Q.\forall x.\mathsf{cellist}(x) \rightarrow Q(x)](\lambda z.\mathsf{smokes}(z))$

This formula, too, can be simplified via two beta reductions. In a first step, we remove λQ and correspondingly replace Q in the value description by $\lambda y.\mathsf{smokes}(y)$. This gives us:

(54) $[\forall x.\mathsf{cellist}(x) \rightarrow [\lambda z.\mathsf{smokes}(z)](x)]$

In a second step, we apply $\lambda z.\mathsf{smokes}(z)$ to x and get $\mathsf{smokes}(x)$. So we end up with:

(55) $\forall x.\mathsf{cellist}(x) \to \mathsf{smokes}(x)$

From here on in, we will not spell out these kinds of beta reductions explicitly.

Exercise 2. Download the Lambda Calculator from `http://lambdacalculator.com`, and install it on your computer. (It works with Mac, Windows and Linux operating systems.) Then open the 'Scratch Pad' and verify for yourself that the two reductions just given work as described.

5.8 Summary

5.8.1 Syntax of L_λ

Let us now summarize our new logic, L_λ, which is a version of the SIMPLY TYPED LAMBDA CALCULUS. The TYPES are defined recursively as follows:

- e is a type
- t is a type
- If σ is a type and τ is a type, then $\langle \sigma, \tau \rangle$ is a type.
- Nothing else is a type.

A FORMULA is an expression of type t.

1. **Basic Expressions**
   For every type τ, L_λ provides a set of NON-LOGICAL CONSTANTS Con_τ, with one constant $c_{n,\tau}$ for each natural number n, and a set of VARIABLES Var_τ, with one variable $v_{n,\tau}$ for each natural number n.

   We use conventional abbreviations for the most common variables: x, y, z for type e; P, Q for type $\langle e, t \rangle$; R for type $\langle e, \langle e, t \rangle \rangle$; p, q for type t. Variables at other types carry an explicit type subscript (e.g., x_a for schema type a). When multiple variable symbols appear in the same formula, they are understood to be distinct from one another — that is, they stand for distinct syntactic variables, not necessarily variables with distinct values. Two typographically distinct variable symbols may be assigned the same value by the assignment function.

2. **Application**
   For any types σ and τ, if α is an expression of type $\langle \sigma, \tau \rangle$ and β is an expression of type σ then $[\alpha(\beta)]$ is an expression of

type τ. (We will often drop the outer square brackets when it does not result in confusion.)

3. **Identity**
   If α and β are expressions of the same type, then $\alpha = \beta$ is a formula (an expression of type t).

4. **Negation**
   If ϕ is a formula, then so is $\neg\phi$.

5. **Binary Connectives**
   If ϕ and ψ are formulas, then so are $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \to \psi]$, and $[\phi \leftrightarrow \psi]$.

6. **Quantification**
   If ϕ is a formula and u is a variable of any type, then $[\forall u.\phi]$ and $[\exists u.\phi]$ are formulas.

7. **Lambda abstraction**
   If α is an expression of type τ and u is a variable of type σ then $[\lambda u.\alpha]$ is an expression of type $\langle\sigma,\tau\rangle$.

Recall that when reading a formula, you may assume that the scope of a binder ($\forall$, $\exists$, or λ) extends as far to the right as possible. So, for example, $\forall x.[P(x) \wedge Q(x)]$ can be rewritten as $\forall x.P(x) \wedge Q(x)$. However, we will typically retain brackets in these cases. Similarly to how we can drop the dot between two quantificational binders, we can also drop the dot between two lambdas in a row, so we can write, e.g. $\lambda x\lambda y.\mathsf{admire}(x,y)$. (The order of the lambdas still matters: this function is not the same as $\lambda y\lambda x.\mathsf{admire}(x,y)$.) We will, however, always retain the final dot in a sequence of lambda binders in order to show that the end of the argument list has been reached, e.g. $\lambda x\lambda y.\exists z.\mathsf{give}(x,y,z)$. Once again, these dot-related conventions are specific to our textbook, and there is no single standard in the field.

To further reduce clutter, we will add the following abbreviatory convention: Square brackets that are immediately embedded

inside parentheses can be dropped. This way, we can for example write $\pi(\lambda x.\mathsf{happy}(x))$ rather than $\pi([\lambda x.\mathsf{happy}(x)])$.

Finally, we define some equivalences between 'relational style' and 'functional style' formulas. For example,

(56) $\mathsf{loves}(x, y)$

is defined to be equivalent to $\mathsf{loves}(y)(x)$. In general, if π denotes an n-place right-to-left curried relation, then

(57) $\pi(\alpha_1)(\alpha_2)...(\alpha_n)$

can be re-written as

(58) $\pi(\alpha_n, \alpha_{n-1}, ..., \alpha_1)$

5.8.2 Semantics of L_λ

As in L_{Pred}, the semantic values of expressions in L_λ depend on a model and an assignment function. As in L_{Pred}, a model M determines a set of individuals, which we will call D, and an interpretation function I that maps non-logical constants of the language to denotations of the appropriate kind based on this set of individuals and the set of truth values {T,F}.

Let $\mathcal{T}$ be the set of types (e for individuals, t for truth values, $\langle e, t\rangle$ for functions from individuals to truth values, etc.). For each type $\tau \in \mathcal{T}$, the model determines a corresponding domain D_τ. Let us define the STANDARD FRAME based on D as an indexed family of sets $(D_\tau)_{\tau\in\mathcal{T}}$, where:

- $D_e = D$
- $D_t = \{\mathsf{T}, \mathsf{F}\}$
- for any types σ and τ, $D_{\langle\sigma,\tau\rangle}$ is the set of functions from D_σ to D_τ.

A MODEL for L_λ based on D, then, is a pair $\langle (D_\tau)_{\tau \in \mathcal{T}}, I \rangle$, where:[4]

- $(D_\tau)_{\tau \in \mathcal{T}}$ is a standard frame based on D
- for every type $\tau \in \mathcal{T}$, I assigns to every non-logical constant of type τ an object from the domain D_τ.

Fundamentally, types are syntactic categories of *expressions* in the logic, but speaking somewhat loosely, we say that τ is the type of an object drawn from D_τ.

Assignments provide values for variables of all types, not just those of type e. An assignment thus is a function assigning to each variable of type τ a denotation from the set D_τ.

The semantic value of an expression is defined as follows:

1. **Basic Expressions**
   (a) If α is a non-logical constant, then $[\![\alpha]\!]^{M,g} = I(\alpha)$.
   (b) If α is a variable, then $[\![\alpha]\!]^{M,g} = g(\alpha)$.
2. **Application**
   If α is an expression of type $\langle \sigma, \tau \rangle$, and β is an expression of type σ, then $[\![\alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$.
3. **Identity**
   If α and β are expressions of the same type, then $[\![\alpha = \beta]\!]^{M,g} = \mathsf{T}$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$.
4. **Negation**
   If ϕ is a formula, then $[\![\neg\phi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{F}$.
5. **Binary Connectives**
   If ϕ and ψ are formulas, then:
   (a) $[\![\phi \wedge \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{T}$ and $[\![\psi]\!]^{M,g} = \mathsf{T}$.
   (b) $[\![\phi \vee \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{T}$ or $[\![\psi]\!]^{M,g} = \mathsf{T}$.

[4] This formalization is inspired by Gallin (1975), p. 12.

(c) $[\![\phi \to \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{F}$ or $[\![\psi]\!]^{M,g} = \mathsf{T}$.

(d) $[\![\phi \leftrightarrow \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = [\![\psi]\!]^{M,g}$.

6. **Quantification**

   (a) If ϕ is a formula and v is a variable of type τ then $[\![\forall v\,.\,\phi]\!]^{M,g} = \mathsf{T}$ iff for all objects $o \in D_\tau$:

   $$[\![\phi]\!]^{M,g[v \mapsto o]} = \mathsf{T}$$

   (b) If ϕ is a formula and v is a variable of type τ then $[\![\exists v\,.\,\phi]\!]^{M,g} = \mathsf{T}$ iff there is some object $o \in D_\tau$ such that:

   $$[\![\phi]\!]^{M,g[v \mapsto o]} = \mathsf{T}$$

7. **Lambda Abstraction**
   If α is an expression of type τ, and u a variable of type σ, then $[\![\lambda u\,.\,\alpha]\!]^{M,g}$ is that function f from D_σ into D_τ such that for all objects o in D_σ, $f(o) = [\![\alpha]\!]^{M,g[u \mapsto o]}$.

5.8.3 Entailment, validity and satisfiability in L_λ

As in $\mathrm{L}_{\mathrm{Pred}}$, denotations in L_λ are computed relative to an assignment as well as a model, so we first fix what it is for a formula to be true in a model. For any expression ϕ of type t, $[\![\phi]\!]^M = \mathsf{T}$ if and only if $[\![\phi]\!]^{M,g} = \mathsf{T}$ for every assignment g, and $[\![\phi]\!]^M = \mathsf{F}$ if and only if $[\![\phi]\!]^{M,g} = \mathsf{F}$ for every assignment g. Where ϕ has no free variables, the choice of g makes no difference.

The notions we have been relying on throughout can now be stated for L_λ as well:

- $\phi_1, \ldots, \phi_n \vDash \psi$ if and only if ψ is true in every model in which all of $\phi_1, \ldots, \phi_n$ are true.
- An argument is VALID if and only if its premises entail its conclusion.

- Two expressions of the same type are EQUIVALENT, written $\alpha \equiv \beta$, if and only if $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$ for every model M and assignment g.
- A formula is SATISFIABLE if and only if it is true in at least one model, and VALID AS A FORMULA if and only if it is true in every model.

These definitions matter because natural language entailment is the data our semantics is meant to predict. Once English sentences are translated into L_λ, a claim about which English sentences entail which becomes a claim about which L_λ formulas stand in the relation $\vDash$.

5.9 Further reading

This chapter has provided just the bare minimum that is needed for starting to do formal semantics. There is very little proof theory in this chapter, and there has been only scant presentation of model theory, so this can hardly be considered a serious introduction to the subject. Carpenter (1998) is an excellent introduction to the logic of typed languages for linguists who would like to deepen their understanding of such issues.

Exercises on typed lambda calculus

5.9.1 Technical exercises

For the following exercises, assume the following typing conventions:

- constants of type e: a, b, c, d, e, f, g
- constants of type $\langle e, t\rangle$: smokes, pirate, sailor, person, french
- variables of type e: x, y, z

- variables of type $\langle e, t\rangle$: P, Q
- variables of type $\langle e, \langle e, t\rangle\rangle$: R

Assume the following lexical entries:

- *likes* $\rightsquigarrow \lambda y.\lambda x.\mathsf{likes}(x, y)$
- *smokes* $\rightsquigarrow \lambda x.\mathsf{smokes}(x)$
- *sailor* $\rightsquigarrow \lambda y.\mathsf{sailor}(y)$
- *pirate* $\rightsquigarrow \lambda y.\mathsf{pirate}(y)$
- *sister* $\rightsquigarrow \lambda x.\lambda y.\mathsf{sister}(x, y)$
- *'s* $\rightsquigarrow \lambda x.\lambda R.\lambda y.R(y)(x)$
- *French* $\rightsquigarrow \lambda x.\mathsf{french}(x)$
- *is* $\rightsquigarrow \lambda P.P$
- *a* $\rightsquigarrow \lambda P.P$
- *everybody* $\rightsquigarrow \lambda Q.\forall x.[\mathsf{person}(x) \rightarrow Q(x)]$
- *every* $\rightsquigarrow \lambda Q.\lambda P.\forall x.[Q(x) \rightarrow P(x)]$
- *Alex* $\rightsquigarrow$ a; *Blake* $\rightsquigarrow$ b; *Casey* $\rightsquigarrow$ c

Exercise 3. Simplifying expressions. After checking that the type of the function and the type of the argument(s) match, simplify the following expressions. If the expression cannot be simplified then just leave it as it is. If the simplification involves multiple steps, show your work by including the intermediate steps in the derivation.

(i) x

(ii) $\mathsf{pirate}(x)$

(iii) $\lambda x.\mathsf{pirate}(x)$

(iv) $\lambda x.[\mathsf{pirate}(x)](a)$

(v) $\lambda x.[\mathsf{pirate}(x)](x)$

(vi) $\mathsf{pirate}(\mathsf{a})$

(vii) $\lambda x.\mathsf{likes}(\mathsf{a},x)$

(viii) $[\lambda y.\lambda x.\mathsf{likes}(x,y)](\mathsf{a})$

(ix) $[[\lambda y.\lambda x.\mathsf{likes}(x,y)](\mathsf{a})](\mathsf{b})$

(x) $\lambda P.P$

(xi) $[\lambda P.P](\mathsf{pirate})$

(xii) $\lambda P.\lambda x.\neg P(x)$

(xiii) $\exists x.\mathsf{sailor}(x)$

(xiv) $\lambda P.\lambda Q.\exists x.[P(x)\wedge Q(x)]$

(xv) $[\lambda P.\lambda Q.\exists x.[P(x)\wedge Q(x)]](\mathsf{sailor})$

(xvi) $[\lambda P.\lambda Q.\exists x.[P(x)\wedge Q(x)]](\lambda x.\mathsf{sailor}(x))$

(xvii) $[\lambda Q.\exists x.[\mathsf{sailor}(x)\wedge Q(x)]](\mathsf{pirate})$

(xviii) $[\lambda Q.\exists x.[\mathsf{sailor}(x)\wedge Q(x)]](\lambda x.\mathsf{pirate}(x))$

(xix) $[\lambda Q.\forall x.[\mathsf{sailor}(x)\rightarrow Q(x)]](\mathsf{pirate})$

(xx) $[\lambda P.[\lambda Q.\forall x.[P(x)\rightarrow Q(x)]]](\mathsf{sailor})$

Exercise 4. Assigning semantic types. Give the semantic type of the following lambda-expressions. You may want to simplify them in your mind or using the scratchpad before assigning a type.

(1) x

(2) a

(3) $\mathsf{pirate}(x)$

(4) $\neg\mathsf{pirate}(x)$

(5) pirate

(6) $\lambda x.\mathsf{pirate}(x)$

(7) $\lambda x.\neg\mathsf{pirate}(x)$

(8) $[\lambda x.\mathsf{pirate}(x)](\mathsf{a})$

(9) $[\lambda x.\mathsf{pirate}(x)](x)$

(10) $\mathsf{pirate}(\mathsf{a})$

(11) $\mathsf{likes}(y,x)$

(12) $\lambda x.\mathsf{likes}(\mathsf{a},x)$

(13) $\lambda y.\lambda x.\mathsf{likes}(x,y)$

(14) $[\lambda y.\lambda x.\mathsf{likes}(x,y)](\mathsf{a})$

(15) $[[\lambda y.\lambda x.\mathsf{likes}(x,y)](\mathsf{a})](\mathsf{b})$

(16) $\lambda P.P$

(17) $[\lambda P.P](\mathsf{pirate})$

(18) $\neg\mathsf{pirate}(x)$

(19) $\lambda P.[\lambda x.P(x)]$

(20) $\lambda P.[\lambda x.\neg P(x)]$

(21) $[\lambda P.\lambda x.\neg P(x)](\text{pirate})$

(22) $\exists x.\text{sailor}(x)$

(23) $\neg\exists x.\text{sailor}(x)$

(24) $\exists x.P(x)$

(25) $\exists x.[\text{sailor}(x) \wedge \text{pirate}(x)]$

(26) $\exists x.[P(x) \wedge Q(x)]$

(27) $\lambda Q.\exists x.[P(x) \wedge Q(x)]$

(28) $\lambda P.[\lambda Q.\exists x.[P(x) \wedge Q(x)]]$

(29) $[\lambda P.[\lambda Q.\exists x.[P(x) \wedge Q(x)]]](\text{sailor})$

(30) $[\lambda P.[\lambda Q.\exists x.[P(x) \wedge Q(x)]]](\lambda x.\text{sailor}(x))$

(31) $[\lambda Q.\exists x.[\text{sailor}(x) \wedge Q(x)]](\text{pirate})$

(32) $[\lambda Q.\exists x.[\text{sailor}(x) \wedge Q(x)]](\lambda x.\text{pirate}(x))$

(33) $\forall x.P(x)$

(34) $P(x)$

(35) $\text{sailor}(x) \rightarrow \text{pirate}(x)$

(36) $\forall x.[\text{sailor}(x) \rightarrow \text{pirate}(x)]$

(37) $[\lambda Q.\forall x.[\text{sailor}(x) \rightarrow Q(x)]]$

(38) $[\lambda Q.\forall x.[\text{sailor}(x) \rightarrow Q(x)]](\text{pirate})$

(39) $[\lambda P.[\lambda Q.\forall x.[P(x) \rightarrow Q(x)]]]$

(40) $[\lambda P.[\lambda Q.\forall x.[P(x) \rightarrow Q(x)]]](\text{sailor})$

5.9.2 Linguistics exercises

Exercise 5. Relational kinship terms like *aunt* can be thought of as denoting binary relations among individuals. We might therefore introduce a binary predicate aunt to represent the aunthood relation, such that a sentence like *Sue is Alex's aunt* could be represented as $\mathsf{aunt}(\mathsf{sue},\mathsf{alex})$. But consider *Sue is an aunt!* (perhaps uttered in a context where Sue's sister just gave birth). This sentence might be taken to express an existential claim like $\exists x.\mathsf{aunt}(\mathsf{sue},x)$. On such a usage, the noun *aunt* might be taken to denote, rather than a binary relation, the property that someone has if there is someone that they are the aunt of: $\lambda y.\exists x.\mathsf{aunt}(y,x)$. In this expression, one of the arguments of the relation is existentially bound. We might imagine that there is a regular process that converts a relational noun like *aunt* into a noun denoting the property of standing in the relevant relation to some individual. Using L_λ, describe a function that would take as input an arbitrary binary relation like the aunthood relation (type $\langle e,\langle e,t\rangle\rangle$) and gives as output the property that an individual has if they stand in this relation to another individual. This is a one-place predicate, so it is of type $\langle e,t\rangle$. The answer should therefore take the form of a lambda expression of type $\langle\langle e,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle$.

Exercise 6. We normally consider *eat* a transitive verb, and according to the kind of analysis we have done here, this would imply a treatment as a binary relation, type $\langle e,\langle e,t\rangle\rangle$. And yet we do have usages where the object does not appear, as in *Have you eaten?* One might imagine that a two-place predicate can be reduced to a one-place predicate through an operation that existentially quantifies over the object argument. Define a function that does this and express it as a well-formed lambda expression

in L_λ. The input to the function should be a binary relation (type $\langle e, \langle e, t \rangle \rangle$) and the output should be a unary relation (type $\langle e, t \rangle$) where the object argument has been existentially quantified over.

Exercise 7. Like *eat*, the verb *shave* can be used both transitively and intransitively; consider *The barber shaved John* and *The barber shaved.* But in contrast to *eat*, the intransitive version does not mean that the barber shaved something; it means that the barber shaved himself. Give an expression of L_λ of type $\langle \langle e, \langle e, t \rangle \rangle, \langle e, t \rangle \rangle$ which produces this sort of denotation from a two-place predicate. (Adapted from Dowty et al. (1981), Problem 4-7, p. 97.)

6 Function Application

6.1 Introduction

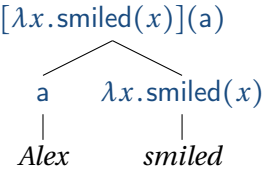
We will now use the lambda calculus to translate constituents of arbitrary size, from words and phrases all the way up to the sentences themselves, into logic. We will show how to carry out a translation of English into the lambda calculus, and how to compose the resulting lambda expressions and their denotations so that the result is a logical formula whose truth conditions are the same as those of the English sentence. Our underlying assumption is that lambda calculus expressions translate syntactic constituents and compose in a way that mirrors the syntactic structure of the sentence. It is the job of a theory of syntax to determine what these constituents are; not just any substring of an English sentence is a constituent. Here we will just give a toy syntax that can be replaced by more sophisticated syntactic theories without significant changes to the semantics. As the rules allow us to derive a meaning for a complex expression from the meanings of the parts, the process by which translations of complex expressions are computed from the translations of their parts is sometimes referred to as a DERIVATION.

How, then, do denotations of constituents compose? We will first explore the hypothesis, inspired by Frege's idea of saturation, that there is only one way for the meanings of two subexpressions to combine to give the meaning of a complex expression: application of a function to an argument. In this chapter, we will define

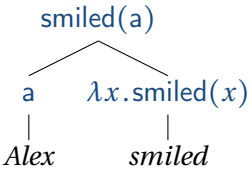
a semantics for a fragment of English that adheres to this principle. To do so, we will *translate* expressions of English into expressions of L_λ. A name like *Alex* will translate as the expression a, which is of type e; both denote the individual Alex. The intransitive verb *smiled* will be translated as the type $\langle e, t\rangle$ expression $\lambda x.\mathsf{smiled}(x)$; both denote the set of smilers. We write the 'translates to' relation as $\rightsquigarrow$:

(1) a. *Alex* $\rightsquigarrow \mathsf{a}$
b. *smiled* $\rightsquigarrow \lambda x.\mathsf{smiled}(x)$

The combination, *Alex smiled*, will then be translated as the result of applying the translation of the verb to the translation of the subject:

(2)


... or equivalently, through beta reduction:

(3)


The formulas at the tops of these trees have the same denotation as each other and as the English sentence *Alex smiled*; that denotation is the truth value of this sentence.

Again, we are using an indirect interpretation method in this book, which means that we translate English to the representation language first (using $\rightsquigarrow$), and then interpret the representation language (using $[\![\cdot]\!]$). So rather than the Heim & Kratzer (1998) style:

(4) $[\![\textit{Alex smiled}]\!] = \text{smiled(a)} = 1$

we instead write:

(5) $\textit{Alex smiled} \rightsquigarrow \mathsf{smiled(a)}$

To express that the sentence is true (relative to a given model M and an assignment function g,) we write:

(6) $[\![\mathsf{smiled(a)}]\!]^{M,g} = \mathsf{T}$

At the time of writing, both styles are widely used in the semantic literature, and the choice depends on what the author finds most convenient for their expository purposes.

We take the denotations of the English expressions to be inherited from those of their translations in lambda calculus.[1] A given sentence can then be said to be true with respect to a model and an assignment function if its translation is true with respect to that model and assignment function.

Indirect interpretation is the style that Montague (1974b) used in his famous work entitled *The Proper Treatment of Quantification in Ordinary English* ('PTQ' for short). There, he specified a set of principles for translating English into a logic. This work stands in contrast to another famous Montague paper, *English as a Formal Language* (Montague, 1974a), in which a direct interpretation style was used. Montague was very clear that this translation procedure was only meant to be a convenience; one *could in principle* specify the denotations of the English expressions directly. So we will continue to think of our English expressions as having denotations, even though we will specify them indirectly via a translation to the lambda calculus. Nevertheless, *the expressions of the lambda calculus are not themselves the denotations, just like the*

[1] Assuming that there may be multiple translations into the representation language for a given expression of English, there is not necessarily a unique denotation, although the representation language is unambiguous. For example, a given word might have multiple distinct translations.

name "Alex" is not itself the person Alex. Lambda calculus expressions are strings with a certain length, structure, etc., while denotations are entities, truth values, sets and functions, etc. We have two languages at play, a natural language such as English (our object language) and the lambda calculus (a formal language, our representation language). We are translating from the natural object language to the formal representation language, and specifying the semantics of the formal representation language in our meta-language (which is also English, mixed with talk of sets and relations).[2]

We will not translate every expression of English to our representation language, only a well-behaved 'fragment' of it, as Richard Montague called it. In 'English as a formal language', Montague (1974a) formally defined the first fragment of English, consisting of the following ingredients: a specification of our formal representation language, with syntactic and semantic rules; a specification of the syntax of the English expressions we cover; a list of

[2] An important difference between the tack we are taking here and the one taken in Heim & Kratzer's (1998) textbook is that here the λ symbol is part of our representation language but not the meta-language, whereas in Heim and Kratzer the λ symbol is part of the meta-language (and there is no distinction between the meta-language and the representation language). For example, in their style, one would write:

(i) $[\![\textit{snores}]\!] = \lambda x\,.\,x$ snores

with a mix of English and lambdas on the right-hand side of the equation. In contrast, we write equations mapping object language to representation language like this:

(ii) $\textit{snores} \rightsquigarrow \lambda x\,.\,\mathsf{snores}(x)$

and equations mapping representation language to denotations specified in the meta-language like this:

(iii) $[\![\lambda x\,.\,\mathsf{snores}(x)]\!]^{M,g} = I(\mathsf{snores})$

One should carefully distinguish between these two ways of using the λ symbol and make sure to be consistent.

lexical entries; and a list of composition rules. Throughout this book we too will build up a fragment in a similar style.

6.2 Syntax

We already have our representation language: L_λ as defined in the previous chapter. The next step is to specify the rules that generate the syntactically well-formed expressions of our fragment of English. We will use a simplistic theory of syntax called context-free grammar. Many details of the syntactic theory don't matter, as long as the syntax delivers the right structure. For example, the syntactic categories we use in the syntax rules and as labels of nonterminals (nodes with daughters) are only for purposes of exposition, and any other set of labels would do just as well.

(7) **Syntax**

S	→	DP VP
VP	→	V (DP|AP|PP)
AP	→	A (PP)
DP	→	D (NP)
NP	→	N (PP)
NP	→	A NP
PP	→	P DP

The vertical bar | separates alternative possibilities, and the parentheses signify optionality, so the VP rule means that a VP can consist solely of a verb, or of a verb followed by an NP, or of a verb followed by an AP, etc.

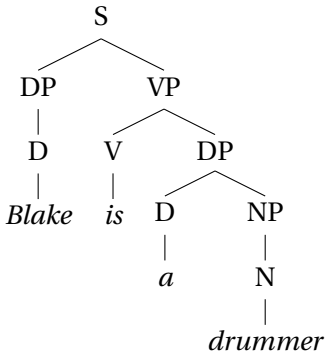
The terminal nodes (nodes without daughters, i.e. leaves) of the syntax trees produced by these syntax rules may be labeled by the following words:

(8) **Lexicon**

V: *smiled, laughed, loves, likes, hugged, is, smokes, drinks*
A: *Swedish, happy, kind, proud*
N: *sailor, pirate, singer, drummer, musician*
D: *a, every, some, no*
D: *Alex, Blake, Casey, everybody, somebody, nobody*
D: *everything, something, nothing*
P: *of, with*

For example, this grammar generates *Blake is a drummer* with the following syntactic structure.

(9)



Exercise 1. Which of the following strings are sentences of the fragment of English that we have defined (modulo sentence-initial capitalization)? Draw syntax trees for those that are. For those that are not, briefly explain why they are not generated by the grammar.

(a) George loves everybody.

(b) Some drummer smiled every happy musician.

(c) Alex is.

(d) No is a happy singer.

(e) Somebody is proud of a singer.

(f) A drummer loves proud of Blake.

(g) A proud drummer of Blake loves every happy happy happy happy drummer.

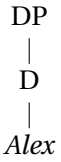
(h) Alex smiles with nobody.

Keep in mind that the syntax might generate sentences that don't make any sense or are not grammatical in English, and that's OK. At least some of the nonsensical sentences will be ruled out once we define semantic interpretations for these words.

In the trees below, sometimes we "prune" non-branching nodes. For example, we might write:

(10)


instead of

(11)


Now that we have defined the syntax of our fragment of English, we need to specify how the expressions generated by these syntax rules are interpreted. To do so, we will translate them into expressions of L_λ. We will associate translations not only with words, but also with syntactic trees. We can think of words as degenerate cases of trees, so in general, translations go from trees to expressions of our logic.

In accordance with Frege's conjecture, at this time we have

only one rule for composing the denotation of a complex expression out of the denotations of the parts. (We will add further rules to our system in later chapters.) Our rule, FUNCTION APPLICATION, just applies a function to an argument:

Composition Rule. Function Application (FA)
Let γ be a syntax tree whose only two subtrees are α and β (in any order) where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type σ.

Then

$$\gamma \rightsquigarrow \alpha'(\beta')$$

(The prime symbol $'$ in α' is not intended to have any meaning of its own; α' is just a convenient way to refer to whatever α is translated as.)

Exercise 2. If γ is a syntax tree whose only two subtrees are α and β (in any order), where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type σ.

then what type does the translation of γ have, assuming that it is translated according to the rule of Function Application?

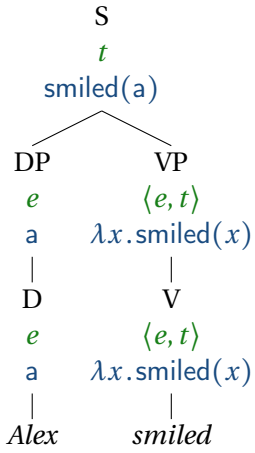
This rule will provide a translation into L_λ for any tree that has two immediate subtrees, as long as their types match appropriately. The node at the top of such a tree is called a BRANCHING NODE because it branches into multiple subtrees. If a tree has

no branches, then it is called a NON-BRANCHING NODE. For non-branching nodes, we will simply assume that the denotation at the higher node is the same as the denotation at the lower one:

Composition Rule. Non-branching Nodes (NN)
If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$, then $\beta \rightsquigarrow \alpha'$.

With these two rules, we can assign denotations to each subtree in the syntactic structure of *Alex smiled* as follows (showing only fully beta-reduced translations at each node, along with their types):

(12)



In order to provide a starting point to the compositional process, we assume that the terminal nodes provided by the syntactic theory each contribute independent semantic values and have translations that are individually stipulated and not determined by rules such as Function Application. What these terminal nodes are can vary from theory to theory. While the traditional picture takes them to be words, certain theories of syntax and morphology identify them in other ways. For example, theories such as Distributed Morphology (Halle & Marantz, 1993) assume that there

is no sharp boundary between word formation and sentence formation; on such theories, the terminal nodes may consist of units that are smaller than a word.

A related question is whether the Function Application rule applies to every branching node or whether it has exceptions. Idioms such as *spill the beans* or *kick the bucket* are often argued to make their semantic contribution to the sentence as a whole. Frameworks such as Construction Grammar (Croft & Cruse, 2004) and Sign-Based Construction Grammar (Boas & Sag, 2012) assume that there is no sharp boundary between the meaning of words and of larger constructions such as idioms; on such theories, one may want to consider these idioms as nonterminal nodes whose translations are not determined by the Function Application rule.

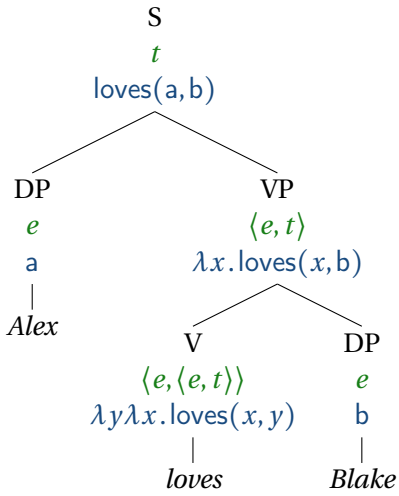
6.3 Combining verbs with arguments

Let us now consider how to analyze a simple transitive sentence like *Alex loves Blake*. We will represent the denotation of the verb *loves* as follows:

(13) $\textit{loves} \rightsquigarrow \lambda y \lambda x . \mathsf{loves}(x, y)$

Can this verb combine semantically with a type-e direct object via Function Application? Yes, it can; the types match. This is shown in the following derivation for *Alex loves Blake*:

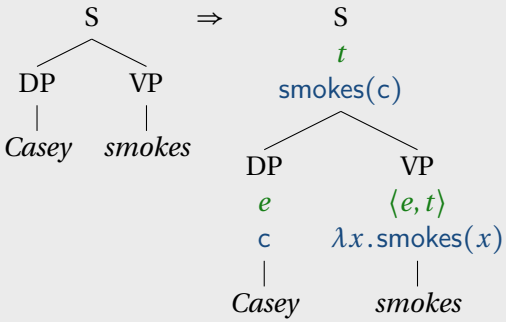
(14)



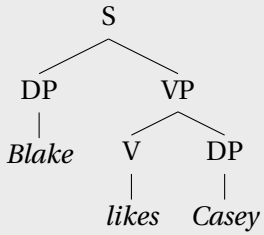
Via Function Application, the transitive verb *loves* combines with the object *Blake*. The VP *loves Blake* thus comes to denote (the characteristic function of) the set of all individuals who love Blake, which we can think of as the property of loving Blake. This property is then attributed to Alex through a second application of Function Application at the top node.

Exercise 3. Practice using Function Application. Give fully beta-reduced translations into L_λ at each node of the tree.

(i) Casey smokes

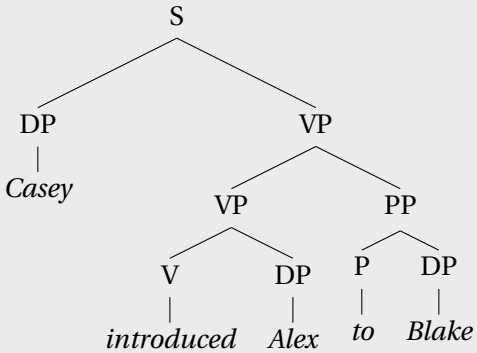


(ii) Blake likes Casey



The first one is done for you. It may help to indicate the type at each node of the tree, as in the sample solution.

Exercise 4. Assume that the ditransitive verb *introduce* is of type $\langle e, \langle e, \langle e, t \rangle \rangle \rangle$. Give a lexical entry for *introduce* of this type and provide appropriate translations for the terminal and nonterminal nodes in the following tree:



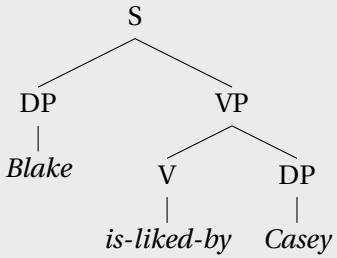
You will also need to assume a lexical entry for *to*. An easy solution is to treat it as an identity function of type $\langle e, e \rangle$.

Exercise 5. Practice with the passive: Give a lexical entry for the

passive verb 'is liked by' and give fully beta-reduced translations at each node of the trees below. For simplicity, you can just treat the three words 'is liked by' as a single word. Make sure that the derivation you end up with at the top requires that Casey likes Blake, and not the other way around.

- *is-liked-by* $\rightsquigarrow$?

(i) Blake is liked by Casey

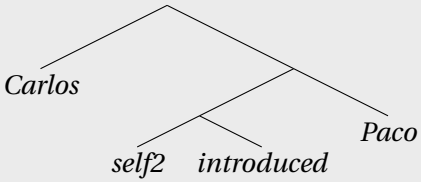


Exercise 6. In some languages, there is a morpheme (e.g., Middle Voice in Ancient Greek, reflexivizing affix in Kannada, Passive Voice in Finnish, etc.) that attaches to the verb stem and reduces its arity by one. Let us take the following imaginary morphemes *self1*, *self2*, and *self3*. Assuming the syntactic structure given, give a denotation for each of these morphemes.

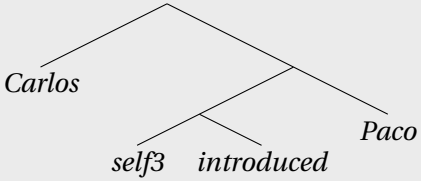
Assume that *Carlos* is an ordinary proper name, translated as a constant of type e, and assume that *shaves* is an ordinary transitive verb, translated as an expression of type $\langle e, \langle e, t \rangle \rangle$. You can use the lexical entry for *introduce* given in the previous exercise.

(a) For the sentence *Carlos self1-shaves*, make the structure below yield the denotation 'Carlos shaves himself' by supplying the denotation of *self1*.

(b) For the sentence *Carlos self2-introduced Paco*, make the structure below yield the denotation 'Carlos introduced Paco to Carlos (himself)' by supplying the denotation of *self2*.



(c) For the sentence *Carlos self3-introduced Paco*, make the structure below yield the denotation 'Carlos introduced Paco to Paco (himself)' by supplying the denotation of *self3*.



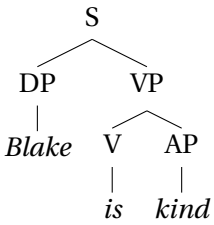
Make sure that your denotations work not just for sentences involving Carlos and Paco, but arbitrary proper names.

(Exercise due to Maribel Romero.)

6.4 Predicative sentences

Now let us consider how to analyze a sentence with an adjective following *is*, such as *Blake is kind*. The syntactic structure is as follows:

(15)



We will continue to assume that the proper name *Blake* is translated as the constant b, of type e. We can assume that *kind* denotes a function of type $\langle e, t\rangle$, the characteristic function of a set of individuals (those that are kind). Let us use kind as a constant of type $\langle e, t\rangle$, and translate *kind* thus.

(16) $\textit{kind} \rightsquigarrow \lambda x.\mathsf{kind}(x)$

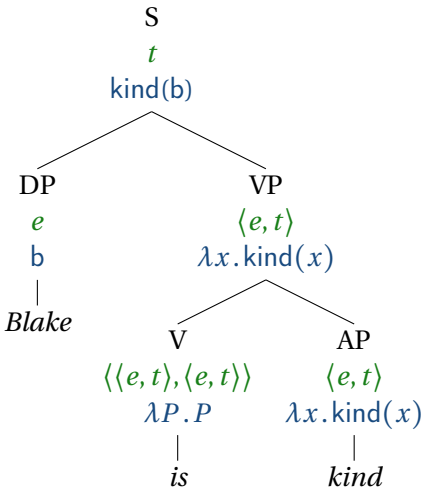
Now, what is the contribution of the copula *is*? Besides signaling present tense, it does not seem to accomplish more than to link the predicate 'kind' with the subject of the sentence. Since we have not started dealing with tense yet, we will ignore the former function and focus on the latter. (We also set aside cases in which *is* indicates identity rather than predication, as in *Mark Twain is Samuel Clemens*.) We can capture the fact that the copula *is* connects the predicate to the subject by treating it as an IDENTITY FUNCTION, a function that returns whatever it takes in as input. In this case, the copula *is* takes in a function of type $\langle e, t\rangle$, and returns that same function. (We adopt the same approach for other words that seem to lack meaning of their own, such as *did* in *Casey did not smile*.)

(17) $\textit{is} \rightsquigarrow \lambda P.P$

This implies that *is* denotes a function that takes as its first argument another function P, where P is of type $\langle e, t\rangle$, and returns P.

With these rules, we will end up with the following analysis for the sentence *Blake is kind*:

(18)



Each node shows the syntactic category, the semantic type, and a fully beta-reduced translation to lambda calculus. In this case, Function Application is used at all of the branching nodes (S and VP), and Non-branching Nodes is used at all of the non-branching non-terminal nodes (DP, V, and AP). The individual lexical entries that we have specified are used at the terminal nodes (*Blake*, *is*, and *kind*).

Like adjectives, prepositional phrases can also serve as predicates, as in, for example, *Alex is with Blake*. Let us translate *with* as follows, invoking a binary predicate with:

(19) $\textit{with} \rightsquigarrow \lambda y \lambda x . \mathsf{with}(x, y)$

Via Function Application, the preposition *with* combines with its object *Blake*, and the resulting PP combines with *is* to form a VP. The translation of the VP is an expression of type $\langle e, t \rangle$, denoting a function from individuals to truth values. This applies to the denotation of *Alex* to produce a truth value.

Like prepositions, adjectives can denote functions of type $\langle e, \langle e, t \rangle \rangle$. *proud* is an example; in *Casey is proud of Alex*, the adjective *proud* expresses a relation that holds between Casey and Alex. We can

capture this by assuming that *proud* translates as:

$$\lambda y \lambda x . \text{proud}(x, y)$$

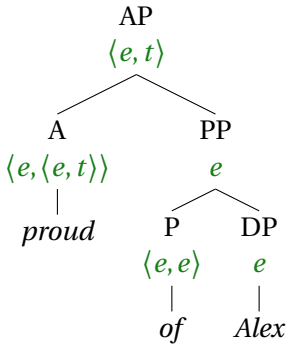
This is an expression of type $\langle e, \langle e, t \rangle \rangle$ denoting a function that takes two arguments, first a potential object of pride (such as Alex), then a potential bearer of such pride (e.g. Casey), and returns True if the pride relation holds between them.

In contrast to *with*, the preposition *of* does not seem to signal a two-place relation in this context. We therefore assume that *of* is a function word like *is*, and also denotes an identity function. Unlike *is*, however, we will treat *of* as an identity function that takes an individual and returns an individual, so it will be of type $\langle e, e \rangle$.

(20) $\quad of \rightsquigarrow \lambda x . x$

So the adjective phrase *proud of Alex* will have the following structure:

(21)



Now let us consider a sentence with a nominal predicate: *Alex is a singer*. The noun *singer* can be analyzed as an $\langle e, t \rangle$ type property like *Swedish*, the characteristic function of the set of individuals who are singers.

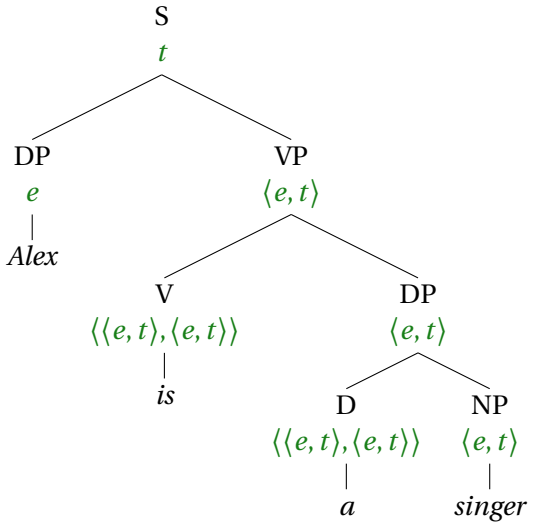
The indefinite article *a* is another function word that appears to be semantically vacuous, at least on its use in the present context. We will assume that *a*, like *is*, denotes a function that takes an

$\langle e, t\rangle$-type predicate and returns it. In general, it is common and convenient to assume that all semantically vacuous words denote such identity functions.

(22) $a \rightsquigarrow \lambda P\,.\,P$

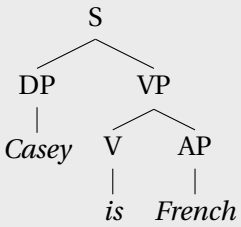
With these assumptions, the derivation will go as follows.

(23)

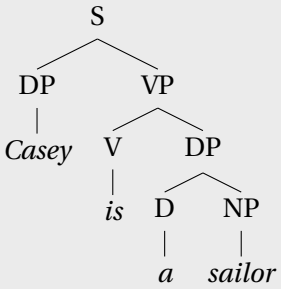


Exercise 7. Provide derivations of the truth conditions for the sentences below using Function Application.

(i) Casey is French



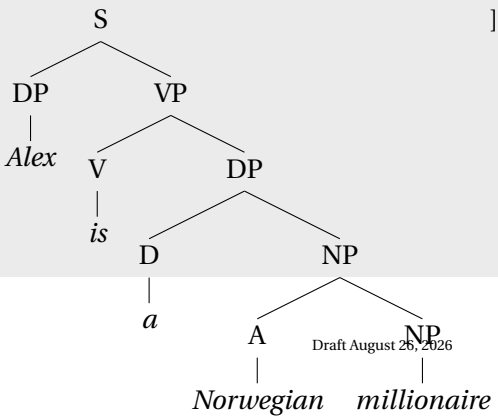
(ii) Casey is a sailor



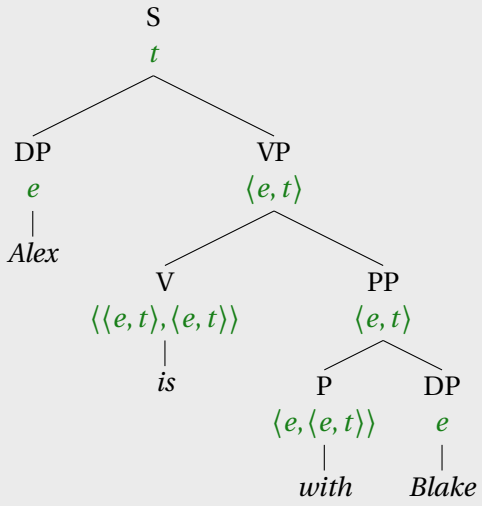
Exercise 8. Give fully beta-reduced translations at each node of the tree for *Alex is a singer*.

Exercise 9. Can the indefinite article *a* be analyzed as $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ in a sentence like *A singer loves Blake*? Why or why not?

Exercise 10. Assume that *Norwegian* and *millionaire* are both of type $\langle e,t\rangle$, following the style we have developed so far. Is it possible to assign truth conditions to the following sentence using those assumptions? Why or why not?



Exercise 11. Derive the translation into L_λ for *Alex is with Blake* by giving a fully beta-reduced translation for each node.

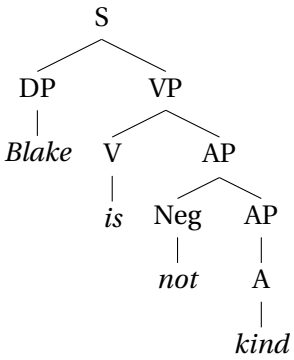


Exercise 12. Give a tree for *Blake is proud of Casey* and a fully beta-reduced form of the translation at each node. (Question to consider: What would go wrong if we were to assume that *of* was of type $\langle e, \langle e, t\rangle\rangle$, like *with*?)

6.5 Introducing negation

Now let us consider how to analyze the word *not* in a sentence like *Blake is not kind*. We will assume the following syntactic structure, where the negative particle *not* adjoins to the adjective phrase to its right:

(24)



Assuming that the negative particle *not* is a word of category Neg, we add the following syntax rule for negation to the list in (7):

(25) Additional syntax rule for negation

$$\text{XP} \quad \rightarrow \quad \text{Neg XP}$$

In this rule, XP is meant to be a variable ranging over any phrasal syntactic category; it could be instantiated for example as AP, PP, DP, or VP. It should be noted that this rule captures some of the syntactic flexibility of negation but also over-generates. For example, our grammar now produces *Casey not smiled*. This sort of structure is fine in many other languages such as German, but in English, verbal negation requires *do*-support, as in *Casey did not smile*. We will gloss over this detail of the morphosyntax of the English auxiliary system, among others, for the sake of simplicity.

The denotation of *Blake is not kind* should be the negation of *Blake is kind*:

(26) $\neg\mathsf{kind}(\mathsf{b})$

Thus the property that *(is) not kind* denotes should be something that applies to an individual and yields 'true' only in the case that the individual is not kind:

(27) $\lambda x.\neg\mathsf{kind}(x)$

The denotation of *not* should apply to a property and produce such a function for any arbitrary predicate, not just *kind*. The following denotation will do the trick:

(28) $not \rightsquigarrow \lambda P \lambda x . \neg P(x)$

This lambda expression denotes a function that takes as input a predicate (P) and returns a new predicate, one that returns True given an input x only if P does *not* hold of x, and otherwise returns False. We will refer to this lambda expression as PREDICATE NEGATION. Note that in this lambda expression, the value description is $\lambda x . \neg P(x)$, so the return value is itself another function.

Exercise 13. Draw a syntax tree for *Blake is not kind*, and give fully beta-reduced translations at each node of the tree.

6.6 Quantification

6.6.1 Quantifiers

Let us now consider how to analyze quantifiers like *everybody* and *nobody*. Consider the sentence:

(29) Everybody smiled.

What is the type of *everybody*? Intuitively, there is no particular individual that it denotes. This already suggests that it is not of type e. As Heim & Kratzer (1998) point out, clear evidence that *everybody* does not have e comes from sentences like the following.

(30) Everybody here is over 30 or everybody here is not over 30.

Compare this sentence to one involving a proper name like *Alex*:

(31) Alex is over 30 or Alex is not over 30.

Example (31) is a tautology, but Example (30) is not.

Suppose that *everybody (here)* denoted a particular individual; call this individual d. Then this sentence would say that d is over 30 or d is not over 30. By the LAW OF THE EXCLUDED MIDDLE (LEM) — the thesis that p or not-p, for any proposition p — this would be a tautology.[3] But (30) is *not* a tautology: it could well be false, for example, in a room with mixed ages. Heim & Kratzer (1998) use this observation to argue that *everybody* does not denote an individual: if it did, LEM would guarantee that (30) is a tautology — but it is not.

Still, clearly *everybody smiled* expresses a proposition that can be true or false, so the sentence as a whole is type t. In particular, the translation of *Everybody smiled* should be something like the following (assuming that every individual in the domain is conceived of as human):

(32) *Everybody smiled* $\rightsquigarrow \forall x.\mathsf{smiled}(x)$

We have assumed that a VP like *smiled* denotes a predicate (type $\langle e,t\rangle$). Informally, then, the contribution of *everybody* to the denotation of a sentence is a template:

(33) $\forall x.\_(x)$

where the verb phrase fills in the underlined slot. This idea can be formally implemented through lambda abstraction. *Everybody* will denote a function that takes an arbitrary predicate P, and yields a truth value: true if everything satisfies P and false if not. The following lexical entry for *everybody* says, "Give me a predicate P as input, and I will return as output a truth value – true if everybody satisfies P, and false otherwise":

[3] LEM should be distinguished from BIVALENCE, which is the property of a logical system that recognizes exactly two truth values. LEM is a theorem about the logical form of propositions; bivalence is a semantic property of the system. They come apart in non-classical logics.

(34) $\textit{everybody} \rightsquigarrow \lambda P.\forall x.P(x)$

(Here we are glossing over the fact that *everybody* quantifies over persons, and just treating it as a synonym of *everything*.)

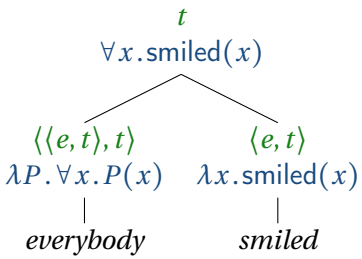
As P is a variable that stands for a predicate—something of type $\langle e,t\rangle$—the type of the expression denoted by *everybody* is:

(35) $\langle\langle e,t\rangle,t\rangle$

This is the type of a QUANTIFIER.

This denotation for *everybody* can be combined via Function Application with the denotation for *smiled* in the following manner:

(36)



In this derivation, the *VP* is fed as an *argument* to the *subject DP*, rather than the other way around. Recall that Function Application does not care about the order of the arguments, so this order of application works just as well as the more familiar situation where the VP takes the subject as an argument.

For any type τ, an expression of type $\langle\tau,t\rangle$ can be seen as a predicate that applies to arguments of type τ. So quantifiers can be seen as higher-order predicates: that is, as predicates of predicates. For instance, *somebody* can be seen as denoting a function that takes as input a predicate and returns true iff there is at least one individual that satisfies the predicate:

(37) $\textit{somebody} \rightsquigarrow \lambda P.\exists x.P(x)$

(Here we are glossing over the fact that *somebody* quantifies over

animate individuals, and just treating it as a synonym of *something*.)

In contrast, the function denoted by *nobody* returns true iff there is nothing satisfying the predicate:

(38) $\textit{nobody} \rightsquigarrow \lambda P . \neg \exists x . P(x)$

(Here again we are glossing over the animacy restriction for *nobody* and treating it as a synonym of *nothing*.)

6.6.2 Quantificational determiners

Now what about determiners like *every*, *no*, and *some*? We want *every singer* to function in the same way as *everyone*, so these should denote functions that take the denotation of a noun phrase and return a quantifier. The input to these determiners (e.g. *singer*) is of type $\langle e, t \rangle$, and their output is a quantifier, of type $\langle \langle e, t \rangle, t \rangle$. So the type of these kinds of determiners will be:

(39) $\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$

In other words, quantificational determiners like *every* expect a predicate (like *singer*) as their argument, and return a function, which itself expects a predicate (like *smiled*). The latter function returns a truth value.

For each of these quantificational determiners, the truth value that is returned depends on the two input predicates, and can be specified using the quantifiers $\exists$ and $\forall$ of first-order logic:

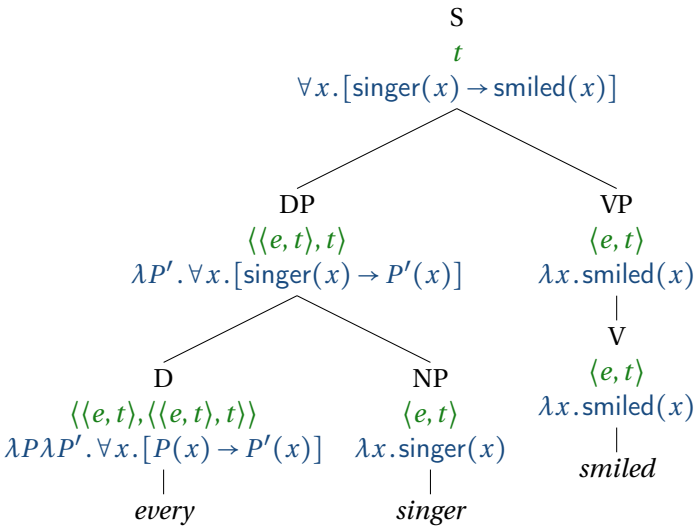
(40) $\textit{some} \rightsquigarrow \lambda P \lambda P' . \exists x . [P(x) \wedge P'(x)]$

(41) $\textit{no} \rightsquigarrow \lambda P \lambda P' . \neg \exists x . [P(x) \wedge P'(x)]$

(42) $\textit{every} \rightsquigarrow \lambda P \lambda P' . \forall x . [P(x) \rightarrow P'(x)]$

These lexical entries will yield analyses like the following:

(43)

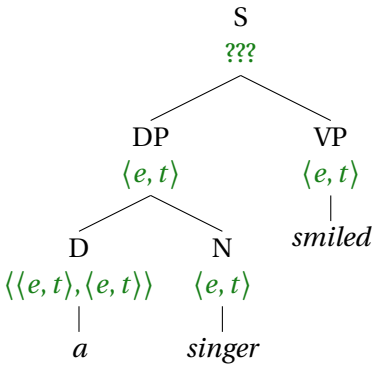


The same strategy can be applied to indefinite descriptions like *a singer*. Previously, we analyzed indefinite descriptions in sentences like *Alex is a singer*, where the indefinite description functions as a PREDICATE, and applies to a subject. But an indefinite description can also function as the subject or object of a transitive verb, as in the following sentences:

(44) a. A singer loves Alex. [subject position]
b. Alex loves a singer. [object position]

In such uses, *a singer* functions as an ARGUMENT of the verb, as opposed to a predicate. (In this instance, we are using the term 'argument' in the sense in which it is used in the study of natural language syntax, referring to a syntactic dependent of an argument-selecting lexical item like a verb.) If we applied our $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ analysis to a case like *A singer smiled*, where the indefinite appears in subject position, we would be in a predicament:

(45)



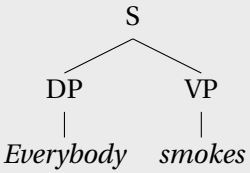
We currently have no rule for combining two expressions of type $\langle e, t \rangle$, because neither is expecting the other as an argument. In the next chapter, we will define a rule that can combine two expressions of type $\langle e, t \rangle$, namely Predicate Modification. But even that rule does not give the right meaning. We can escape this predicament by providing the indefinite article with a translation as a quantificational determiner.

Exercise 14. Give an analysis of *A singer loves Alex* using Function Application and Non-Branching Nodes. Your analysis should take the form of a tree, specifying at each node, the syntactic category, the semantic type, and a fully beta-reduced translation to L_λ. The translation of the sentence should be true in any model where there is some individual that is both a singer and someone who loves Alex. You may have to introduce a new lexical entry for the indefinite article *a*. Your analysis should account for the fact that the sentence is true in any model where there is an individual who is both a singer and who stands in the 'loves' relation with Alex, and no others. Note that such an entry would have to be an *additional* entry, not a replacement: the predicative entry $\lambda P . P$ is still needed for cases like *Alex is a singer*, where *a singer* functions as a predicate rather than as an argument. Indefinite descriptions thus appear to have both a predicative and a quantificational use,

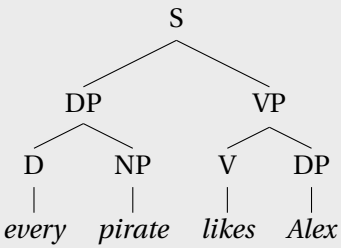
a question we set aside here.

Exercise 15. Give fully beta-reduced translations at each node of the trees below.

(i) Everybody smokes

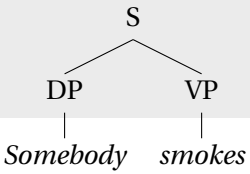


(ii) Every pirate likes Alex

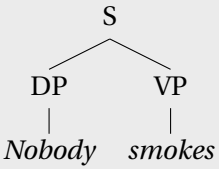


Exercise 16. Using the denotations given above for *no*, *nobody*, *some*, and *somebody*, give fully beta-reduced translations at each node of the trees below.

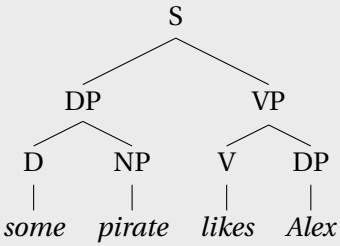
(i) Somebody smokes



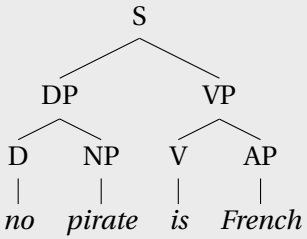
(ii) Nobody smokes



(iii) Some pirate likes Alex



(iv) No pirate is French



6.6.3 Negation and Quantifiers

Consider the English sentence *Everyone doesn't like Sam.* What truth conditions does our current theory derive for this? If we use the predicate negation marker presented in (28) and the lexical entry for *everybody* that we've just given, then we derive truth conditions on which the universal quantifier scopes over negation: for every person, that person does not like Sam, as shown in (46)

below.

(46) $\forall x.\neg \mathsf{likes}(x, \mathsf{s})$

That is indeed the most natural reading out of the blue. But context and prosody can bring out another reading:

(47) Speaker 1: Everyone likes Sam!
Speaker 2: Well, EVERYONE doesn't like Sam. (Jim doesn't like him.)

In this case, negation scopes over the entire sentence, including the quantifier, as shown in (48) below.

(48) $\neg \forall x.\mathsf{likes}(x, \mathsf{s})$

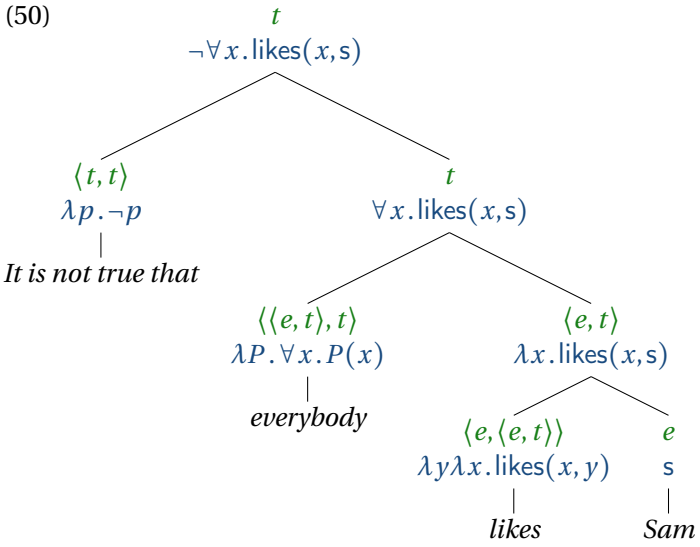
One way of accounting for this ambiguity is to posit two different meanings for negation. The first is **predicate negation**, which was presented in (28). Predicate negation negates the predicate in a subject-predicate structure, as in *Bjorn is not kind*. The second is **sentential negation**. In this case, *not* operates at the sentence level.

A phrase in English that unambiguously expresses sentential negation is *It is not true that*, which can be analyzed as a function of type $\langle t, t \rangle$, flipping true to false and false to true:

(49) *It is not true that* $\rightsquigarrow \lambda p.\neg p$

Let us imagine that English has a lexical item of category 'SNeg' (for 'sentential negation') which is pronounced *it is not true that*. A derivation for *It is not true that everybody likes Sam* is shown in the following tree.

(50)



To account for the reading of *Everybody doesn't like Sam* where *not* scopes over *everybody*, let us assume that *not* has a sentential negation meaning along with its predicate negation meaning:

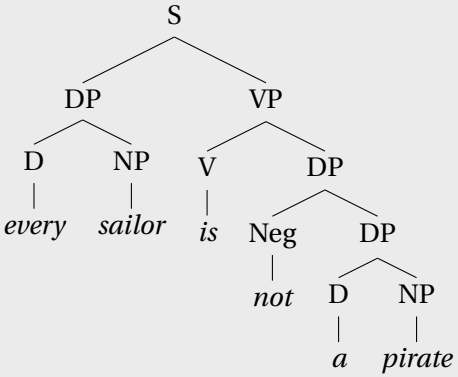
(51) $not \rightsquigarrow \lambda p.\neg p$

By eta reduction, $\lambda p.\neg p$ is equivalent to $\neg$ itself, so this entry simply gives *not* the meaning of the negation operator of our representation language.

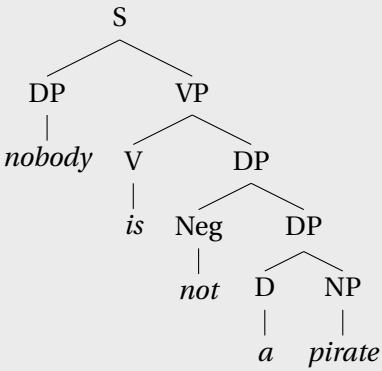
Of course, syntactically, at least on the surface, *not* combines with a verb phrase rather than a sentence. But appearances can be deceiving. As we discuss in more detail in Chapter 7, theories of grammar commonly make a distinction between at least two levels of syntactic representation, which can be called Surface Structure (SS) and Logical Form (LF). The latter is the level of syntactic representation that is actually the input to the semantics. If we assume that at LF, *not* appears higher in the tree than *everybody* as in (50), then we can derive the inverse scope reading for *Everybody doesn't like Sam*.

Exercise 17. Combining quantification and negation.

(i) Every sailor is not a pirate



Exercise 18. Give fully beta-reduced translations at each node of the following tree for *Nobody is not a pirate.* What reading do you derive? Give a plain-English paraphrase of the derived truth conditions.



In Standard English, two negative elements always yield a **double-negation (DN)** reading, as this derivation illustrates.

Many other languages instead allow two negative elements to express a single negation—a phenomenon called NEGATIVE CONCORD. The typology of negative concord and a compositional analysis are developed in the Appendix to this chapter (p. 231).

6.6.4 Quantifiers in object position

So far, we have restricted our attention to sentences with quantifiers in subject position, as in *Somebody likes Sam.* We have not considered sentences with quantifiers in object position, as in *Sam likes somebody.* It is not possible to give an analysis of the latter type of sentences using Function Application alone, because the types of the transitive verb and the quantificational object do not align for FA. We will return to the Problem of Quantifiers in Object Position in Chapter 7, where we develop two analyses, each more general than the one given here: one using Quantifier Raising (QR) and Predicate Abstraction, and one using type-shifting. For now, we introduce a TYPE-SHIFTING OPERATION that resolves the type mismatch directly.

The combination fails because neither expression is a function that takes the other's type as its argument: the transitive verb has type $\langle e, \langle e, t\rangle\rangle$, which is a function from individuals to predicates, not a function from generalized quantifiers to anything; and the generalized quantifier has type $\langle\langle e, t\rangle, t\rangle$, which takes a predicate as input, not a relation.

The solution is a TYPE-SHIFTING OPERATION: an operation that takes an expression of one semantic type and produces a semantically related expression of a different type, making composition by Function Application possible. The type-shifting operation that resolves the mismatch between a transitive verb and a quantificational object is called RAISE-O ('Object Raising').

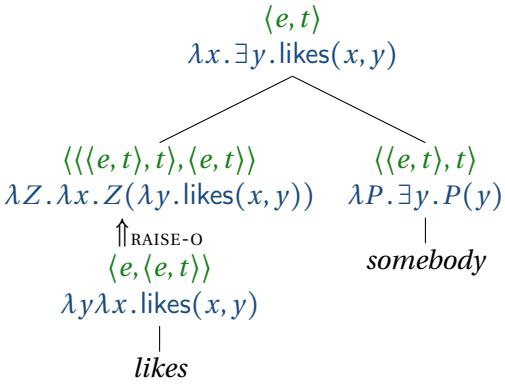
RAISE-O applies to a transitive verb $V \rightsquigarrow R$ of type $\langle e, \langle e, t\rangle\rangle$ and produces a new expression of type $\langle\langle\langle e, t\rangle, t\rangle, \langle e, t\rangle\rangle$:

$$\textsc{raise-o}(V) = \lambda Z.\lambda x.Z(\lambda y.R(y)(x))$$

The shifted verb is now a function from generalized quantifiers (type $\langle\langle e,t\rangle,t\rangle$) to VP-type predicates (type $\langle e,t\rangle$), so Function Application can apply directly between the shifted verb and the quantificational object.

Following the convention we will adopt systematically in Chapter 7, the shift is shown inside the verb's node: the shifted denotation and type appear at the top, separated from the base denotation and type at the bottom by the $\Uparrow_{\textsc{raise-o}}$ arrow. The derivation for *likes somebody* becomes:

(52)



The VP denotation $\lambda x.\exists y.\textsf{likes}(x,y)$ is derived in two steps: first RAISE-O shifts the verb to type $\langle\langle\langle e,t\rangle,t\rangle,\langle e,t\rangle\rangle$, then FA applies between the shifted verb and *somebody*.

Sentences with quantifiers in object position are sometimes scopally ambiguous as well. Consider for example:

(53) Sam doesn't like everybody.

This can mean either 'It is not the case that Sam likes everybody' (on which reading Sam still might like someone), or 'Everybody is such that Sam doesn't like them', in other words, 'Sam likes nobody'. On the former reading, negation takes scope over the uni-

versal quantifier, and on the latter, the universal quantifier takes scope over negation.

Exercise 19. (a) Use predicate negation and the type-shifting operation RAISE-O to derive a representation for (53). Which scope reading do you derive, negation over universal or universal over negation? (b) Now use sentential negation instead of predicate negation. Which scope reading do you derive now? (c) Does the theory we have developed so far account for the scope ambiguity in (53)?

Exercise 20. Using the type-shifting operation RAISE-O and the lexical entry for *nobody* given above, compositionally derive a meaning representation for *likes nobody*. Show the RAISE-O step explicitly in your tree.

6.7 Generalized quantifiers

The formal properties of natural language determiners — conservativity, the weak/strong distinction, and the cardinal/proportional distinction — are developed in Chapter 2, Section 2.4.2. The connection between those properties and the behavior of NEGATIVE POLARITY ITEMS (NPIs) such as *ever* and *any* is explored in Chapter 2.

6.8 Toy fragment

The following is a summary of the toy fragment developed in this chapter.

Representation language.
This chapter uses L_λ, the simply-typed lambda calculus.

Type conventions.

Variables of type e: x, y, z

Variables of type $\langle e,t\rangle$: P, Q

Variables of type $\langle\langle e,t\rangle,t\rangle$: Z

Variables of type $\langle e,\langle e,t\rangle\rangle$: R

Variables of type t: p, q

Constants of type e: a, b, c

Constants of type $\langle e,t\rangle$: smiled, laughed, smokes, drinks, kind, swedish, happy, proud, sailor, pirate, singer, drummer, musician, person

Constants of type $\langle e,\langle e,t\rangle\rangle$: loves, likes, hugged, with

Variables of types not listed above carry an explicit type subscript (e.g., x_a for a variable of schema type a). When multiple variable symbols appear in the same formula, they are understood to be distinct from one another.

Syntax.

S	→	DP VP
S	→	SNeg S
VP	→	V (DP|AP|PP)
AP	→	A (PP)
DP	→	D (NP)
NP	→	N (PP)
NP	→	A NP
PP	→	P DP

Schemas (where *XP* ranges over any phrasal category):

$$XP \quad \rightarrow \quad \text{Neg } XP$$

Composition rules.

- **Function Application** (FA)
  Let γ be a tree whose only two subtrees are α and β where:
  - $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
  - $\beta \rightsquigarrow \beta'$ where β' has type σ.

  Then
  $$\gamma \rightsquigarrow \alpha'(\beta')$$

- **Non-branching Nodes** (NN)
  If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$, then $\beta \rightsquigarrow \alpha'$.

Type-shifting operations.

- **RAISE-O** (Object Raising)
  Applied to a transitive verb $V \rightsquigarrow R$ of type $\langle e, \langle e, t \rangle \rangle$, produces an expression of type $\langle \langle \langle e, t \rangle, t \rangle, \langle e, t \rangle \rangle$:
  $$\text{RAISE-O}(V) = \lambda Z . \lambda x . Z(\lambda y . R(y)(x))$$
  The shifted verb then combines with a quantificational DP of type $\langle \langle e, t \rangle, t \rangle$ by Function Application to produce a VP of type $\langle e, t \rangle$.

Lexical entries.

V
smiled, laughed, smokes, drinks follow $W \rightsquigarrow \lambda x . W(x)$; *hugged* follows $V \rightsquigarrow \lambda y \lambda x . V(x, y)$.

- *is* $\rightsquigarrow \lambda P . P$
- *loves* $\rightsquigarrow \lambda y \lambda x . \mathsf{loves}(x, y)$
- *likes* $\rightsquigarrow \lambda y \lambda x . \mathsf{likes}(x, y)$

A
Swedish, happy, kind, proud follow $W \rightsquigarrow \lambda x.\, W(x)$.

N
sailor, pirate, singer, drummer, musician follow $W \rightsquigarrow \lambda x.\, W(x)$.

D (proper names)

- *Alex* $\rightsquigarrow$ a
- *Blake* $\rightsquigarrow$ b
- *Casey* $\rightsquigarrow$ c

D (determiners)

- *a* $\rightsquigarrow$ $\lambda P.\, P$
- *some* $\rightsquigarrow$ $\lambda P \lambda Q.\, \exists x.\, [P(x) \wedge Q(x)]$

  Note: the entries for *a* and *some* are strikingly different, even though both are indefinite determiners. The difference reflects the fact that *a* is listed here in its predicative use, as in *Alex is a singer*, whereas *some* has only a quantificational use — compare the unacceptable **Alex is some singer*.
- *no* $\rightsquigarrow$ $\lambda P \lambda Q.\, \neg \exists x.\, [P(x) \wedge Q(x)]$
- *every* $\rightsquigarrow$ $\lambda P \lambda Q.\, \forall x.\, [P(x) \rightarrow Q(x)]$

D (quantificational noun phrases)

- *somebody* $\rightsquigarrow$ $\lambda P.\, \exists x.\, [\mathsf{person}(x) \wedge P(x)]$
- *nobody* $\rightsquigarrow$ $\lambda P.\, \neg \exists x.\, [\mathsf{person}(x) \wedge P(x)]$
- *everybody* $\rightsquigarrow$ $\lambda P.\, \forall x.\, [\mathsf{person}(x) \rightarrow P(x)]$
- *something* $\rightsquigarrow$ $\lambda P.\, \exists x.\, P(x)$

- *nothing* $\rightsquigarrow \lambda P.\neg\exists x.P(x)$
- *everything* $\rightsquigarrow \lambda P.\forall x.P(x)$

P

- *of* $\rightsquigarrow \lambda x.x$
- *with* $\rightsquigarrow \lambda y\lambda x.\mathsf{with}(x,y)$

Neg

- *not* $\rightsquigarrow \lambda P\lambda x.\neg P(x)$

SNeg

- *It is not true that* $\rightsquigarrow \lambda p.\neg p$

The toy fragment is deliberately small, and a number of phenomena that feature prominently in natural language fall outside its scope. For instance, the fragment does not include the definite article *the*. A noun phrase formed with *the*, such as *the singer*, is called a DEFINITE DESCRIPTION. Giving *the* an appropriate semantics requires the notion of presupposition: the definite article presupposes that a unique individual satisfies the restrictor predicate, a kind of meaning that goes beyond truth conditions alone. Definite descriptions are taken up in Chapter 8. Possessive constructions such as *Alex's friend* are also absent; an analysis is developed in an Appendix to this chapter (p. 243).

The fragment can be extended to model cross-linguistic variation in the expression of negation. In negative concord languages, two negative elements express a single negation rather than double negation, requiring a different semantics for negative indefinites. This extension is developed in an Appendix to this chapter (p. 231). Treating quantificational noun phrases as expressions of type $\langle\langle e,t\rangle,t\rangle$ — the generalized quantifier semantics developed here — also illuminates the distribution of NEGATIVE POLARITY

ITEMS (NPIs) such as *ever* and *any*, via the notion of downward entailment. This connection was explored in Chapter 2.

Several further phenomena are deferred to later chapters. Chapter 7 introduces Predicate Modification, which allows two expressions of type $\langle e, t \rangle$ to combine (as in the interpretation of *kind singer*), and develops a more general treatment of quantifiers in object position using Quantifier Raising and Predicate Abstraction. Chapter 9 introduces coordination and plurals. Chapter 10 introduces event semantics, which can be used to model various sorts of VP modifiers. Tense and aspect are treated in Chapter 11. The semantic analysis of modal expressions and intensionality is the topic of Chapter 12.

Appendix: Negative Concord

NEGATIVE CONCORD (NC) is a cross-linguistically widespread phenomenon in which two formally negative elements combine to express a *single* negation. English and Swedish behave as expected: each negative element contributes its own negation, so two negative elements yield a double-negation (positive) reading. Russian and Spanish behave differently: a negative indefinite and a verbal negation marker can co-occur while still expressing only a single negation. This appendix first surveys the cross-linguistic data, then develops a compositional fragment for Swedish, and finally examines the theory of negative concord proposed by Zeijlstra (2007).

Part 1: Cross-linguistic patterns

The paradigm below presents key sentences in English, Swedish, Spanish, and Russian. Sentences labeled (SN) express a *single negation*; sentences labeled (DN) express *double negation*.

Partial affirmation (subject):

(54) a. Someone slept. English
b. Någon sov. Swedish
c. Alguien durmió. Spanish
d. Kto-to spal. Russian

Full negation (subject):

(55) a. Nobody slept. English (SN)
b. Ingen sov. Swedish (SN)
c. Nadie durmió. Spanish (SN)
d. **Nikto ne** spal. Russian (SN, verbal neg. *ne* obligatory)
e. *Nikto spal. Russian (*without *ne*)

Full negation (object):

(56) a. Greta sees nobody. / Greta doesn't see anybody. English (SN)
b. Greta ser ingen. Swedish (SN)
c. Greta **no** ve a **nadie**. Spanish (SN, verbal neg. *no* required)
d. Greta **nikogo ne** vidit. Russian (SN, *ne* obligatory)

Double negation (subject):

(57) a. Nobody didn't sleep. English (DN — awkward)
b. Ingen sov inte. Swedish (DN — awkward)
c. ??Nadie no durmió. Spanish (degraded)

Exercise 1. Based on the data in (54)–(57), answer the following questions.

(a) Which of the four languages exhibits negative concord? What is the relevant evidence? In your answer, define what negative concord means in terms of the relationship between form (number of negative morphemes) and meaning (number of negations expressed).

(b) Which languages do *not* exhibit negative concord? What data support this classification?

Exercise 2. Spanish and Russian are both NC languages, but they differ. Spanish *nadie* can appear in pre-verbal subject position *without* overt verbal negation and still yield a single-negation reading (55)c. Russian *nikto* is ungrammatical without the verbal negation marker *ne* (55)e. Zeijlstra (2007) classifies Spanish as a NON-STRICT NC language and Russian as a STRICT NC language.

(a) Describe in your own words the difference between strict and non-strict NC languages, using Spanish and Russian as examples.

(b) Which sentence in (55) shows that Spanish is non-strict rather than strict? Explain.

Part 2: Swedish

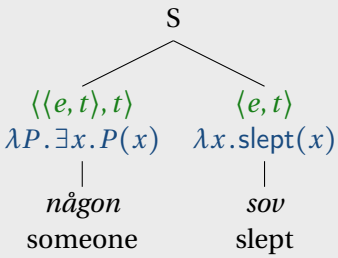
Swedish is a non-NC language that patterns with English, making it a useful starting point. Each negative element contributes its own negation, so two negative elements yield a double-negation (positive) reading. The relevant Swedish items and their translations are:

(58)
a. *någon* (someone): $\rightsquigarrow \lambda P.\exists x.P(x)$ type $\langle\langle e,t\rangle,t\rangle$
b. *ingen* (nobody): $\rightsquigarrow \lambda P.\neg\exists x.P(x)$ type $\langle\langle e,t\rangle,t\rangle$
c. *inte* (not): $\rightsquigarrow \lambda P\lambda x.\neg P(x)$ type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
d. *sov* (slept): $\rightsquigarrow \lambda x.\mathsf{slept}(x)$ type $\langle e,t\rangle$
e. *ser* (sees): $\rightsquigarrow \lambda y\lambda x.\mathsf{see}(x,y)$ type $\langle e,\langle e,t\rangle\rangle$

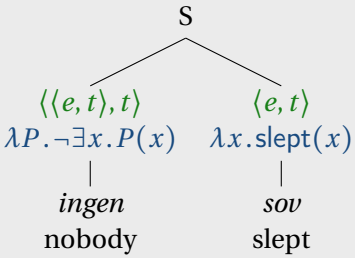
As elsewhere in this chapter, the entries for *någon* and *ingen* gloss over the fact that these expressions quantify over animate individuals; the animacy restriction plays no role in the negative concord facts at issue here, so we omit the person condition that the corresponding English entries carry.

Exercise 3. Complete the following derivations for Swedish sentences with subject-position quantifiers. Fill in the translation at the root node and show the fully beta-reduced result.

(i) *Någon sov* 'Someone slept':



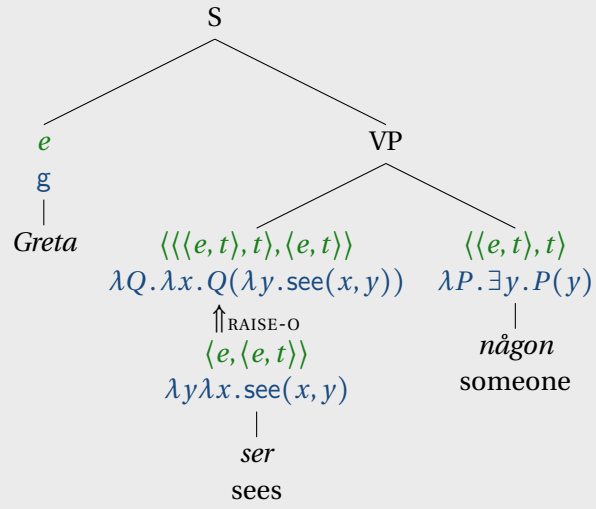
(ii) *Ingen sov* 'Nobody slept':



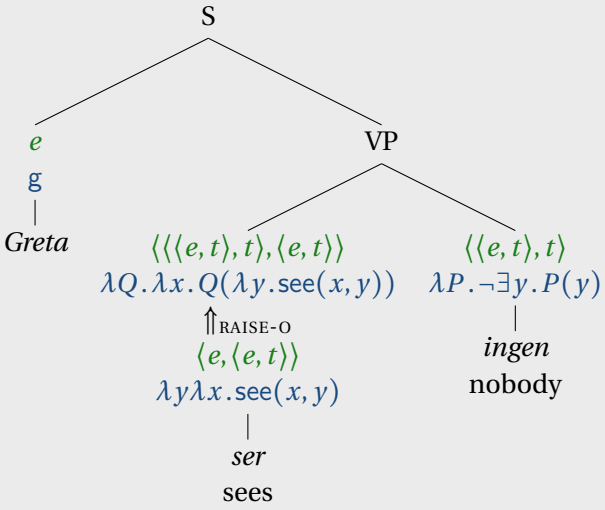
For each sentence, state the derived truth conditions in plain English.

Exercise 4. Complete the following derivations for Swedish.

(i) *Greta ser någon* 'Greta sees someone':

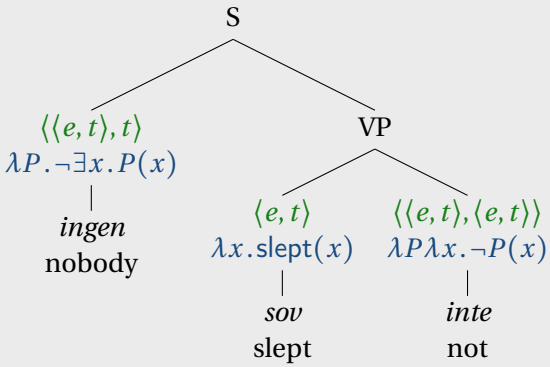


(ii) *Greta ser ingen* 'Greta sees nobody':



For each sentence, state the derived truth conditions in plain English.

Exercise 5. Complete the derivation below for the Swedish sentence *Ingen sov inte* 'Nobody didn't sleep', in which a negative-quantifier subject co-occurs with verbal negation.



Is the derived reading a single-negation (full negation) reading or a double-negation reading? State the truth conditions in plain English.

Part 3: Zeijlstra's theory of negative concord

Zeijlstra (2007) proposes that all negative elements—both verbal negation markers and negative indefinites—carry a NEG FEATURE, which may be either INTERPRETABLE ([iNEG]) or UNINTERPRETABLE ([uNEG]), following the terminology of Minimalist syntax.

- An element bearing [iNEG] is *semantically negative*: it contributes negation as part of its lexical semantics.

- An element bearing [uNEG] makes *no independent contribution to semantic negation*; its NEG feature must be *licensed* by a c-commanding [iNEG] element. Negative indefinites bearing [uNEG] (e.g. *nikto*, *nadie*) translate as existential quantifiers. Verbal negation markers bearing [uNEG] (e.g. Russian *ne*) are semantically vacuous.

When no overt [iNEG] element is present to license a [uNEG] item, a silent operator $\mathbf{OP}_{\text{iNEG}}$, translating as $\lambda p.\neg p$ (type $\langle t,t\rangle$), is inserted above the clause. This operator may only be inserted when the tree contains an unlicensed [uNEG] feature that requires it.

The feature specifications and lexical entries for Russian and Spanish are as follows.

Russian:

(59) a. $ne_{[\text{uNEG}]} \rightsquigarrow \lambda P\lambda x.P(x)$ type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
(semantically vacuous; agree with $\mathbf{OP}_{\text{iNEG}}$)

b. *nikto*/*nikogo*$_{[\text{uNEG}]} \rightsquigarrow \lambda P.\exists x.P(x)$ type $\langle\langle e,t\rangle,t\rangle$
(existential)

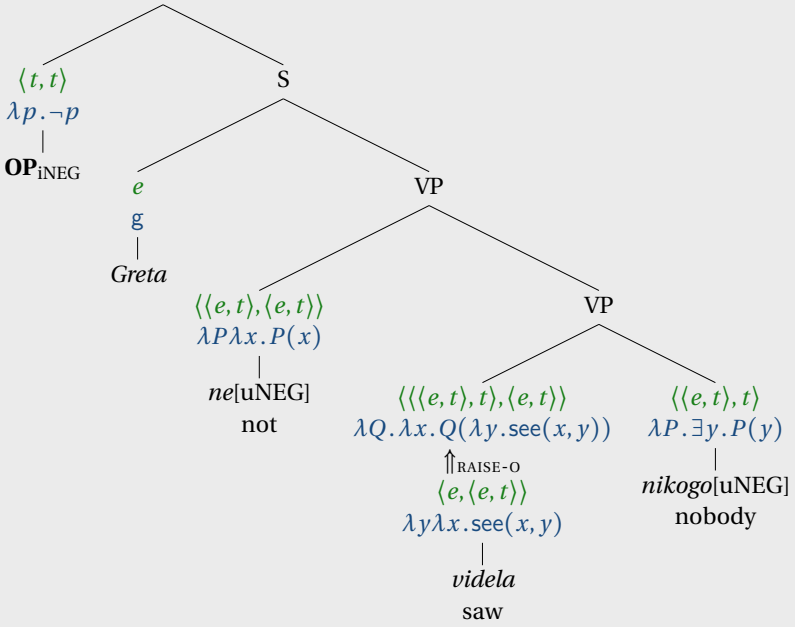
Spanish:

(60) a. $no_{[\text{iNEG}]} \rightsquigarrow \lambda P\lambda x.\neg P(x)$ type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
(predicate negation)

b. *nadie*$_{[\text{uNEG}]} \rightsquigarrow \lambda P.\exists x.P(x)$ type $\langle\langle e,t\rangle,t\rangle$
(existential)

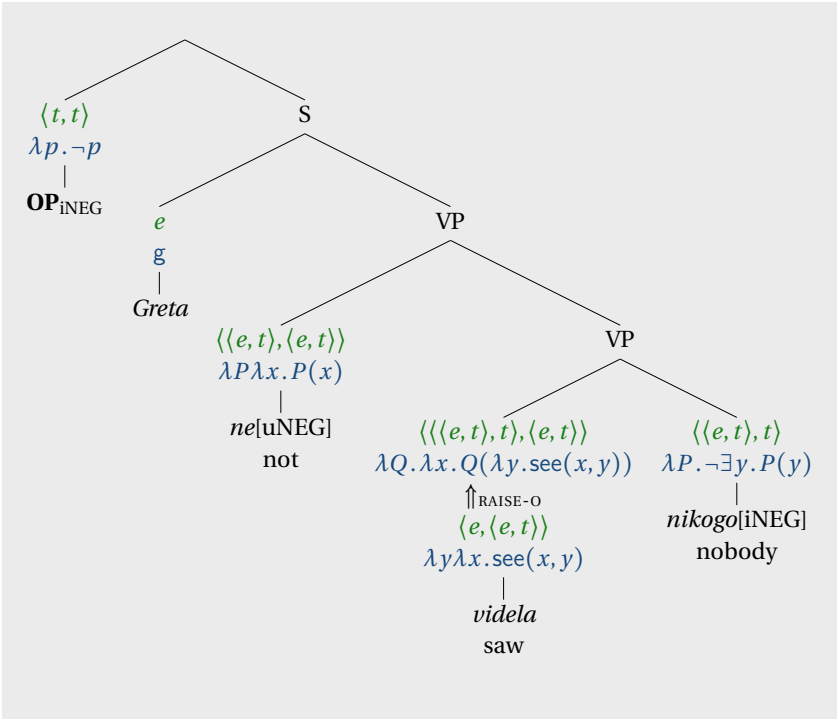
Exercise 6. Use the Zeijlstra lexical entries for Russian given above, together with the type-shifting operation RAISE-O, to complete the following derivations. Fill in all blank nodes.

(i) *Greta ne videla nikogo* 'Greta didn't see anyone' (SN):



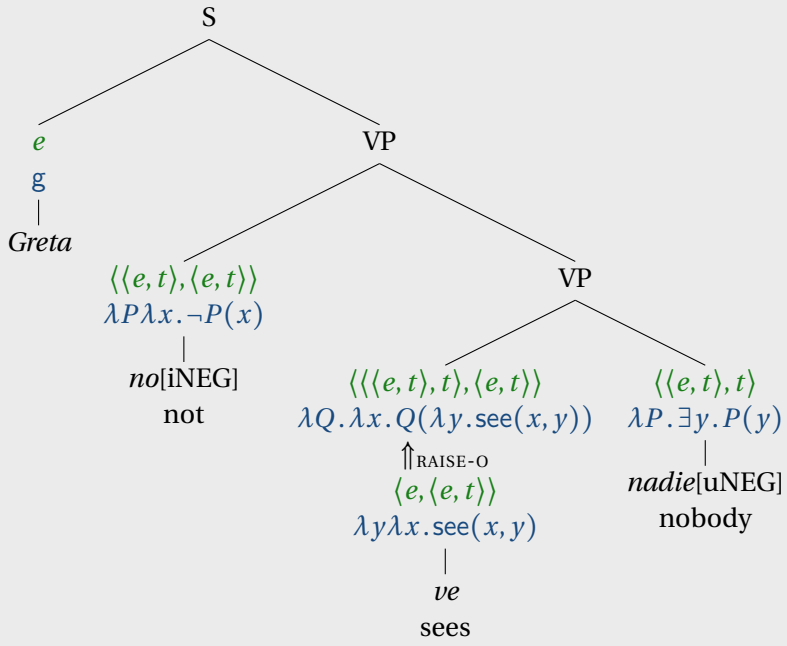
What reading do you derive? Why does this give a single-negation result even though *nikogo* is a negative indefinite?

(ii) Now suppose, contrary to Zeijlstra, that *nikogo* bore [iNEG] and translated as $\lambda P.\neg\exists y.P(y)$. Complete the following derivation and state what reading results. What does this show about why the [uNEG] analysis of *nikogo* is preferable?


⟨t, t⟩
λp.¬p
OP iNEG
S
e
g
Greta
VP
⟨⟨e, t⟩, ⟨e, t⟩⟩
λPλx.P(x)
ne[uNEG]
not
VP
⟨⟨⟨e, t⟩, t⟩, ⟨e, t⟩⟩
λQ.λx.Q(λy.see(x, y))
RAISE-O
⟨e, ⟨e, t⟩⟩
λyλx.see(x, y)
videla
saw
⟨⟨e, t⟩, t⟩
λP.¬∃y.P(y)
nikogo[iNEG]
nobody

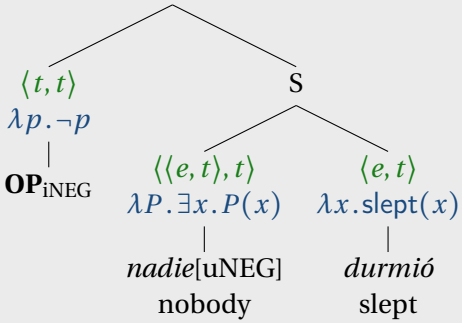
Exercise 7. Use the Zeijlstra lexical entries for Spanish, together with RAISE-O where needed, to complete the following derivations. Fill in all blank nodes.

(i) *Greta no ve a nadie* 'Greta doesn't see anyone' (SN). The overt [iNEG] element *no* licenses *nadie*; no silent operator needed.



What reading results? Why is no silent $\mathbf{OP}_{\text{iNEG}}$ needed here?

(ii) *Nadie durmió* 'Nobody slept' (SN). In pre-verbal subject position, *nadie* bears an unlicensed [uNEG] feature. Zeijlstra inserts the silent $\mathbf{OP}_{\text{iNEG}}$ above the clause to serve as the licensor.



(a) Complete the derivation. What reading results?

(b) Why can the silent operator be inserted here? Why could it *not* be inserted in part (i) above, even though *nadie* also appears there?

(c) Compare this derivation with the Spanish object-position case in (i). In both cases the ultimate reading is a single negation. What is the structural difference between the two derivations?

Exercise 8. Drawing on your work in the previous exercises, consider how Zeijlstra's theory organizes the cross-linguistic variation observed in Part 1.

(a) Fill in the table below with the NEG feature ([iNEG] or [uNEG]) that Zeijlstra assigns to each element in Russian and Spanish.

	Verbal neg marker (*ne* / *no* / *inte*)	Negative indefinite (*nikto* / *nadie* / *ingen*)
Russian		
Spanish		
Swedish		

(b) What do Russian and Spanish have in common at the *theoretical* level, according to Zeijlstra? What property distinguishes NC languages from non-NC languages like English and Swedish?

(c) At the descriptive level, Russian *nikto* and Swedish *ingen* both mean 'nobody' and both yield a single-negation reading in subject position. How does Zeijlstra's theory explain why they behave differently in object position and in co-occurrence with verbal negation?

(d) What is the difference between strict NC (Russian) and non-strict NC (Spanish) in Zeijlstra's framework? Why can Spanish *nadie* appear in pre-verbal subject position without overt verbal negation, while Russian *nikto* cannot?

Appendix: Possessives

Possessive constructions like *Blake's sister* or *Blake's car* raise an interesting compositional puzzle. For some nouns—so-called RELATIONAL NOUNS such as *sister*, *teacher*, or *brother*—the possessive relationship is lexically encoded in the noun itself, and the meaning of the construction follows directly from Function Application. For others—SORTAL NOUNS such as *car*, *house*, or *mansion*—the relationship expressed by the possessive is contextually supplied, and straightforward FA runs into a type mismatch. This appendix develops analyses of both cases and examines two competing strategies for handling sortal nouns.

The following syntax rules and lexical category extend the Toy Fragment (Section 6.8):

DP	→	DP[GEN] NP
DP[GEN]	→	DP Poss

Poss: *'s*. The semantics of *'s* is worked out in the exercises below.

Part 1: Relational nouns

A relational noun like *sister* expresses a two-place relation between individuals. We assign it type $\langle e, \langle e, t\rangle\rangle$:

(61) a. *sister*: $\rightsquigarrow \lambda y.\lambda x.\mathsf{sister}(y)(x)$ type $\langle e, \langle e, t\rangle\rangle$

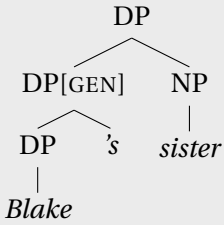
Intuitively, $\mathsf{sister}(y)(x) = 1$ iff x is a sister of y. The target meaning for *Blake's sister* is the property $\lambda x.\mathsf{sister}(\mathsf{b})(x)$—the set of Blake's sisters.

Exercise 1. Assume that *Blake's sister* has the syntactic structure [DP [DPGEN [DP *Blake*] *'s*] [NP *sister*]], and that the possessive marker *'s* produces the intended meaning via Function Application alone. Treat proper names as type-e constants: *Blake* $\rightsquigarrow \mathsf{b}$.

(a) What must the type of *'s* be? Reason through the input and output types: consider what type the DPGEN node must have in order to combine with the noun via FA, and then determine what type *'s* must have in order to combine with the possessor.

(b) Provide a denotation for *'s* consistent with this type.

(c) Complete the derivation tree below for *Blake's sister*, filling in the translation and type at every non-terminal node.



Part 2: Sortal nouns

A sortal noun like *car* picks out a set of individuals; it has type $\langle e, t\rangle$:

(62) *car*: $\rightsquigarrow \lambda x.\mathsf{car}(x)$ type $\langle e, t\rangle$

The meaning of the noun phrase *Blake's car* can be represented as follows:

(63) $\lambda x.\mathsf{car}(x) \wedge R(\mathsf{b})(x)$

where R is a free variable of type $\langle e, \langle e, t\rangle\rangle$ whose value is supplied by context (e.g. the ownership relation, the loan relation, and so on).

Exercise 2. Consider applying the same *'s* from Exercise 1 to the possessed sortal noun *Blake's car*, using the structure [DP [DPGEN [DP *Blake*] *'s*] [NP *car*]].

(a) At which node does the derivation go wrong?

(b) What specifically goes wrong? Show the type mismatch explicitly.

Part 3: Two strategies

Two main strategies have been proposed to extend the possessive analysis to sortal nouns (see Partee, 1983/1997; Vikner & Jensen, 2002).

Strategy (i) — Ambiguity. Posit a second lexical meaning for *'s* that is specifically designed for sortal nouns. This version of *'s* directly introduces the free relation variable R, pairing the possessor with the sortal property.

Strategy (ii) — Type-shift. Keep the single *'s* from Exercise 1, but add a type-shifting operation FREE-R that converts a sortal noun (type $\langle e, t\rangle$) into a relational noun (type $\langle e, \langle e, t\rangle\rangle$) by pairing the property with a free contextual relation R:

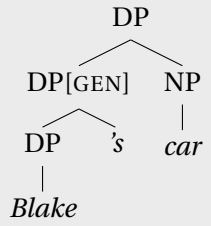
(64) FREE-R: $\rightsquigarrow \lambda P.\lambda y.\lambda x.[P(x) \wedge R(y)(x)]$

The shifted noun then has type $\langle e, \langle e, t\rangle\rangle$, and the original *'s* applies to it exactly as in the relational case.

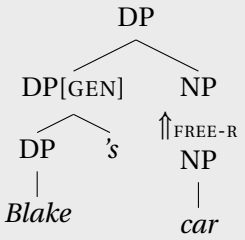
Exercise 3. The trees below show both strategies applied to *Blake's car*. Complete each derivation, filling in the translation and type

at every non-terminal node. After completing both trees, confirm that the resulting denotations are equivalent and state the truth conditions in plain English.

(i) *Ambiguity strategy*: *'s* has a distinct denotation for sortal nouns. Work out what that denotation must be.



(ii) *Type-shift strategy*: *car* is type-shifted by FREE-R; the relational *'s* from Exercise 1 then applies.



Part 4: "Mary's former mansion" (extra credit)

Partee (1983/1997) observes that the two strategies make different predictions for possessives involving the non-intersective adjective *former*. The adjective *former* is not subsective: *Mary's former mansion* does not entail that anything is currently a mansion, so the former mansions are not a subset of the mansions. (The same is true of *fake*: a fake mansion is not a mansion.) Its denotation is:

(65) a. *former*: $\rightsquigarrow \lambda P.\lambda x.\mathsf{formerly}(P(x))$ type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$

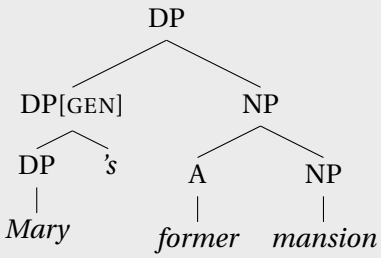
where formerly is a propositional operator of type $\langle t,t\rangle$.

Treating formerly as an operator of type $\langle t,t\rangle$ is a simplification. With just two truth values there are only four functions from truth values to truth values — the identity function, negation, and the two constant functions — and formerly is none of them: whether something was formerly a mansion does not depend only on whether it is one now. In this sense *former* is not extensional, and a proper treatment requires the intensional machinery of Chapter 12. We set that complication aside here, since what is at issue is the relative order in which *former* and the possessive relation apply.

The key question is whether *former* applies to *mansion* (the bare sortal noun) *before* or *after* the possessive relation R is introduced. The two strategies force different answers: under the ambiguity strategy *former* always modifies the sortal noun directly, while under the type-shift strategy the order in which FREE-R and *former* apply can vary.

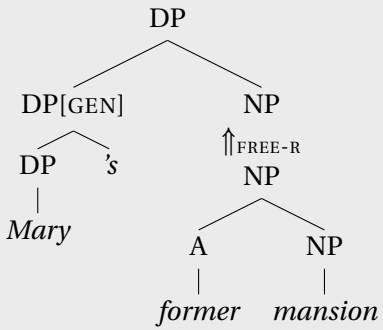
Exercise 4. The two trees below correspond to the two strategies applied to *Mary's former mansion*. Complete each derivation, filling in the translation and type at every non-terminal node.

(i) *Ambiguity strategy* / Partee 1983/1997: *former* combines with *mansion* before the possessive marker applies.



(ii) *Type-shift strategy* / Vikner & Jensen 2002: FREE-R type-shifts

the NP formed by *former* + *mansion* before *'s* applies.



(a) What reading does each tree produce? Describe each in plain English. Are they the same or different?

(b) Vikner & Jensen (2002) show that the type-shift strategy allows an alternative placement of FREE-R, in which FREE-R is applied to *mansion* first, and then *former* is lifted by a type-shift REL—with denotation $\lambda f.\lambda R.\lambda y.f(R(y))$ and type $\langle\langle\langle e,t\rangle,\langle e,t\rangle\rangle,\langle\langle e,\langle e,t\rangle\rangle,\langle e,\langle e,t\rangle\rangle\rangle\rangle$—to operate on the result. Sketch this tree. What reading does it yield, and how does it differ from the reading in (ii)?

7 Beyond Function Application

7.1 Introduction

In the previous chapter, we built up a first compositional theory of semantics for a fragment of English, using only one composition rule: Functional Application. This chapter continues in the same vein, but we will add new composition rules: Predicate Modification, Predicate Abstraction, and the Pronouns and Traces rule.

Let us begin with an example. At the 2013 trial of economist Vicky Pryce, the wife of former British Energy secretary Chris Huhne, the jury asked the judge, Justice Sweeney, the following question:

(1) Can you define what is reasonable doubt?

Justice Sweeney replied:

(2) A reasonable doubt is a doubt which is reasonable.

That this reply does not seem informative was probably part of the judge's point.[1] But for us, the reply does reveal something about the entailment patterns that adjectives like *reasonable* give rise to.

In (2), the adjective *reasonable* appears twice: in ATTRIBUTIVE position before the noun *doubt*, and in PREDICATIVE position after the auxiliary verb *is*. Sweeney seems to imply that one can reason

[1] `https://www.bbc.com/news/uk-21521460`. Retrieved October 7, 2019. He went on to say: "These are ordinary English words that the law doesn't allow me to help you with beyond the written directions that I have already given."

from the attributive to the predicative use, and vice versa:

(3) This is a reasonable doubt. (Premise, attributive use)
∴ This doubt is reasonable. (Conclusion, predicative use)

(4) This doubt is reasonable. (Premise, predicative use)
∴ This is a reasonable doubt. (Conclusion, attributive use)

A related, but distinct point is that dropping this adjective when it occurs in attributive position also results in a valid argument:

(5) This is a reasonable doubt. (Premise)
∴ This is a doubt. (Conclusion)

We will call adjectives like *reasonable* INTERSECTIVE ADJECTIVES: they denote sets, and the noun phrase they modify denotes the intersection of those two sets. Not all adjectives are intersective, however. The following argument is not valid:

(6) Einstein is an outstanding physicist.
Einstein is a violinist.
∴ Einstein is an outstanding violinist.

Adjectives like *outstanding* are SUBSECTIVE: *outstanding physicist* denotes a subset of the set of physicists, but not their intersection with any fixed set of "outstanding things." Dropping the adjective is still valid (an outstanding physicist is a physicist), but the cross-predicate inference above is not.

Some phrases appear to be ambiguous between an intersective and a non-intersective reading. A famous example is *beautiful dancer*:

(7) Nureyev is a beautiful dancer.

On the intersective reading, this sentence is equivalent to saying that Nureyev is beautiful and is a dancer (but not necessarily one who dances beautifully). On the non-intersective reading, this is equivalent to saying that Nureyev dances beautifully (but is not

necessarily beautiful in other respects). Other examples of the same kind include *old* (as in *old friend*) and *big* (as in *big idiot*).

Yet other adjectives are neither intersective nor subsective. This includes adjectives like *alleged*, *former*, *wannabe*, *counterfeit*, and *fake*. For example, the following reasoning is not valid:

(8) John is an alleged murderer.
∴ John is a murderer.

The set of alleged murderers will typically include some murderers but not only murderers, so it is not a subset of the set of murderers.[2]

Among our goals in this chapter will be to expand our fragment of English in a manner that allows us to capture the entailments that various types of adjectives give rise to. As we will see, the composition rule that we introduce for intersective adjectives (Predicate Modification) will also be applicable to relative clauses as in *the representative who Sandy called*. But confronting relative clauses will lead us to develop an additional composition rule (Predicate Abstraction) that can be applied more broadly, in particular to the analysis of quantifiers in object position. The rest of this chapter takes on each of these topics in turn.

Exercise 1. *Frida is a former millionaire* does not entail *Frida is a millionaire* and **Frida is former*. In this sense, *former* is a non-intersective modifier. Which of the following are non-intersective modifiers? Give examples to support your point.

(a) *deciduous*

[2] A further subclass of non-subsective adjectives — including *counterfeit*, *fake*, and perhaps *former* — are called PRIVATIVE: they seemingly map sets to disjoint sets, so that no fake gun is a real gun. Whether this holds depends on precisely what the noun denotes, since sentences like *this gun is fake* and *is this gun real or fake?* suggest that the noun can apply to fake guns too. We set this issue aside.

(b) *presumed*

(c) *future*

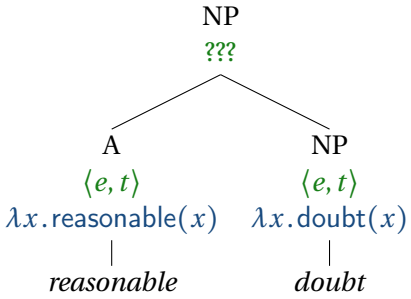
(d) *good*

(e) *mere*

7.2 Predicate Modification

Consider again the sentence *This is a reasonable doubt.* In this sentence, the two expressions *reasonable* and *doubt* are sisters in the tree, but neither one denotes a function that has the denotation of the other in its domain, so Function Application cannot be used to combine them. So far, we have no other rules that could be of use. A situation like this is called a TYPE MISMATCH. Type mismatches occur when two sister nodes in a tree have denotations that are not of the right types for any composition rule to combine them.

(9)



One option for handling this sort of case is to expand our suite of composition rules with a special rule for intersective modification.

The logical counterpart of intersection is conjunction. From

the conjunction $[A \wedge B]$ we can reason to A and to B and back. (Here we translate the demonstrative *This* with a constant this, as if it was a proper name. We do this for convenience only and it should not be taken too seriously. We discuss the semantics of demonstratives a bit more in Chapter 11.)

(10) This is a reasonable doubt.
$[\text{reasonable}(\text{this}) \wedge \text{doubt}(\text{this})]$

(11) This is reasonable.
$\text{reasonable}(\text{this})$

(12) This is a doubt.
$\text{doubt}(\text{this})$

Using conjunction to translate sentences with attributively-used intersective adjectives will explain their entailment patterns.

The rule of Predicate Modification implements this idea. (Intersective Modification would perhaps be a more fitting name.) It takes two predicates of type $\langle e, t \rangle$, and combines them into a new predicate also of type $\langle e, t \rangle$. The new predicate holds of anything that satisfies both of the old predicates:

Composition Rule. Predicate Modification (PM)
If:

- γ is a tree whose only two subtrees are α and β
- $\alpha \rightsquigarrow \alpha'$
- $\beta \rightsquigarrow \beta'$
- α' and β' are of type $\langle e, t \rangle$

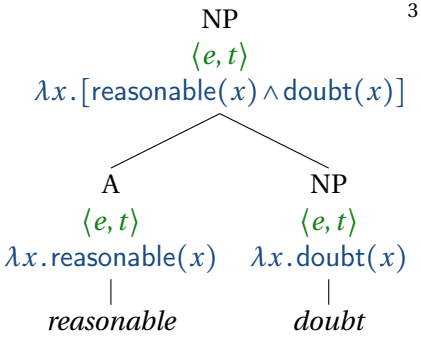
Then:

$$\gamma \rightsquigarrow \lambda u.[\alpha'(u) \wedge \beta'(u)]$$

where u is a variable of type e that does not occur free in α' or β'.

This gives us the following analysis for the NP *reasonable doubt*:

(13)



[3]

Exercise 2. Consider the sentence *John is a vegetarian farmer*. Give a compositional derivation of the truth conditions of this sentence using Predicate Modification. Give your analyses in the form trees that show for each node, the syntactic category, the type, and a fully beta-reduced translation.

Subsective adjectives like *outstanding* and non-subsective adjectives like *alleged* can be given translations of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ and analyzed via Function Application rather than Predicate Modification; the subsectivity entailment is then encoded as a MEANING POSTULATE. We do not develop these cases formally here.

[3] An equivalent result is achievable without a new composition rule, via a type-shifting rule (MOD) that maps an adjective α' of type $\langle e,t\rangle$ to $\lambda P.\lambda x.[\alpha'(x) \wedge P(x)]$ of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$, allowing it to combine with the noun via Function Application. We write $\Uparrow_{\text{NAME}}$ on a tree node to indicate that a type-shifting rule has applied; this notation is used in Section 7.4.1. Nothing in what follows depends on choosing PM over MOD.

7.3 Variable binding at LF

7.3.1 Relative clauses

We turn now to another construction that uses the rule of Predicate Modification, namely relative clauses. Recall that Judge Sweeney defined the construction in (14a), which involves an attributive use of the adjective *reasonable*, in terms of (14b), which involves a predicative use of the same adjective:

(14) a. reasonable doubt
b. doubt which is reasonable

The expression *which is reasonable* is a relative clause. Both *reasonable* and *which is reasonable* serve to restrict the set of doubts under consideration to a subset that are reasonable. Suppose we assume that *which is reasonable* denotes a set: the set of reasonable things. Then, it can combine via Predicate Modification with *doubt* to produce an expression that is equivalent to *reasonable doubt*.

Other relative clauses can be treated as set-denoting expressions as well. Consider:

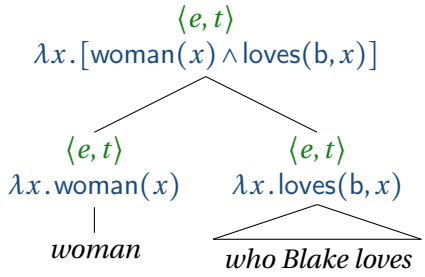
(15) woman who Blake loves

This expression characterizes any individual who has the following two properties: (i) she is a woman; (ii) Blake loves her. In other words, this expression denotes (the characteristic function of) the intersection between the set of women and the set of individuals that Blake loves. Such an interpretation can be derived compositionally if we assume that the relative clause *who Blake loves* is translated as an expression of type $\langle e, t \rangle$:

$$\lambda x.\mathsf{loves}(\mathsf{b}, x)$$

woman is translated as $\lambda x.\mathsf{woman}(x)$. Since both *woman* and *who Blake loves* translate to expressions of type $\langle e, t \rangle$, they can

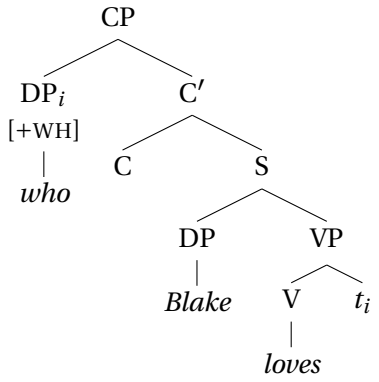
combine via Predicate Modification, like so:



This expression captures the fact that a *woman who Blake loves* is both a woman and an individual loved by Blake.

The question now becomes how we can compositionally derive translations like this for relative clauses. As we have seen, the verb *loves* is transitive, so in ordinary, so-called 'canonical' sentences of English, this verb is followed by an object. But in this case, the relative pronoun *who*, which intuitively corresponds to the object of the verb, appears at the left edge of the relative clause *who Blake loves*.

One way of understanding the connection between *who* and the object of *loves* is by assuming that there are (at least) two levels of syntactic representation, one where *who* occupies the canonical object position immediately following the verb (the 'Deep Structure' of 1960s Chomskyan syntax), and another where it has moved to its so-called 'surface position' (the 'Surface Structure'). Under this view, *wh-* words like *who* (along with *which*, *where*, *what*, etc.) do not disappear entirely from their original positions; they leave a TRACE signifying that they once were there. (Contemporary theories of syntax often use the term UNPRONOUNCED COPY for a related notion that plays essentially the same role for purposes of semantics.) The syntactic structure of the relative clause after movement would then be:



The subscript *i* on *who* represents an INDEX, which allows us to link the *wh-* word to its base position. It can be instantiated as any natural number, such as 1, 3, or 47, so long as it is the same as that of the trace. The element t_i is a TRACE of movement, and because the *wh*-word and the trace bear the same index, we say that the two expressions are CO-INDEXED. It is the job of syntax, rather than semantics, to ensure that all relative pronouns are co-indexed with their traces.

The category label CP stands for 'Complementizer Phrase', because it is the type of phrase that can be headed by a complementizer in relative clauses (see below). The *wh-* word occupies the so-called 'specifier' position of CP (sister to C′).[4] In this structure, the C position is thought to be occupied by a silent version of the complementizer *that*. We hear the complementizer *that* instead of the relative pronoun *who* in, for example, *woman that Blake loves*.[5]

[4]The term 'specifier' comes from the X-bar theory of syntax, where all phrases are of the form [$_{XP}$ (specifier) [$_{X'}$ [$_{X}$ (complement)]]]. See for example Carnie (2013, Ch. 6).

[5]One reason to think that the word *that* is not of the same category as relative pronouns such as *who* or *which* is that only relative pronouns participate in so-called 'pied-piping':

(i) a. good old-fashioned values [$_{\text{CP}}$ on *which* we used to rely]

To explain the fact that *that* and *which* cannot co-occur in English, we assume that either the relative pronoun or the complementizer *that* is deleted, in accordance with the 'Doubly-Filled Comp Filter' (Chomsky & Lasnik, 1977), the principle that either the relative pronoun or *that* must be silent in English.[6]

The same kind of movement is thought to occur in a relative clause like *who likes Alex* or *which is reasonable*, in which it is the subject, rather than the object, that is extracted. In such relative clauses, the trace occurs in subject position:

---

b. *good old-fashioned values [$_{CP}$ on *that* we used to rely]

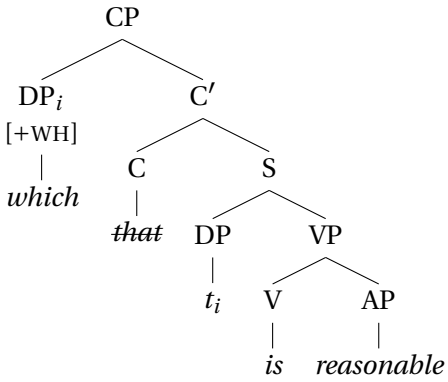
This contrast can be understood under the assumption that *which* originates as the complement of *on*, and moves together with it, while *that* is generated in its surface position. Furthermore, the complementizer *that* is not found only in relative clauses; it also serves to introduce other finite clauses, as in *John thinks that Mary came*. Moreover, in some languages, relative pronouns can actually co-occur with complementizers (Carnie, 2013, Ch. 12). One example is Bavarian German (Bayer, 1984, p. 24):

(ii) I woaß ned **wann dass** da Xavea kummt.
I know not **when that** the Xavea comes
'I don't know when Xavea is coming'

The possibility of their co-occurrence provides additional evidence for the idea that relative pronouns like *who* and complementizers like *that* occupy distinct positions in relative clauses.

[6] See Carnie (2013, Ch. 12) for a more thorough introduction to the syntax of relative clauses.

(16)



In this tree, the relative pronoun *which* is co-indexed with a trace in the subject position for the embedded auxiliary verb *is*.[7] Because the movement changes only the underlying structure and not the sequence of words that is pronounced, this kind of movement is called STRING-VACUOUS MOVEMENT.

These syntactic assumptions lay the groundwork for a semantic treatment of relative clauses on which they function much like adjectival modifiers. The key assumptions are the following:

- Relative clauses are formed through a movement operation that leaves a trace.
- Traces are translated as variables.
- A relative clause is interpreted by introducing a lambda operator that binds this variable.

Which variable does a trace like t_3 correspond to? Recall that in L_λ we have an infinite number of variables in stock. For every natural number i and every type τ, we have a variable of the form v_i. A trace may in principle correspond to a variable of any type.

[7] While the trace theory is widely used in linguistics at the time of writing, some minimalist theories of movement postulate unpronounced copies instead of traces (Fox, 2002). For a recent proposal on how to integrate this "copy theory" of movement with compositional semantics, see Pasternak (2020).

But in the cases we are considering at the moment, it works best to assume that the traces are of type e.

In the compositional system we are setting up here, a trace with a given index always will be translated as a variable of type e with the same index. For example, the trace t_7 would be interpreted as x_7:

$$t_7 \rightsquigarrow x_7$$

This technique will allow the trace and the associated relative pronoun to be linked up in the semantics, as we will choose a matching variable for the lambda expression to bind when we reach the co-indexed relative pronoun in the tree.

The denotation of the variable x_7 will then depend on an assignment; recall from our definition of the semantics of variables in L_λ that:

$$[\![x_7]\!]^{M,g} = g(x_7)$$

Since traces are translated as variables, and variables are interpreted using assignment functions, traces ultimately get their denotation from assignment functions.[8]

We have thus arrived at a new composition rule:

Composition Rule. Pronouns and Traces Rule
If α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow x_i$

[8] Contrast Heim & Kratzer's (1998) rule, given in a direct interpretation style, where an assignment function decorates the denotation brackets: $[\![\alpha_i]\!]^g = g(i)$. Here the difference between direct and indirect interpretation becomes bigger than mere substitution of square brackets for squiggly arrows: In indirect interpretation, we translate pronouns and traces as logical variables. Note that the meta-language still contains its own variables in Heim and Kratzer's style, and these can be bound by lambda operators, as in $[\![\text{loves him}_i]\!]^g = \lambda x\,.\,x \text{ loves } g(i)$. Here, variables like x appear on the right-hand side and variables like 'him$_i$' appear on the left-hand side.

We have called it the 'Pronouns and Traces Rule' because it will also be used for pronouns; for example:

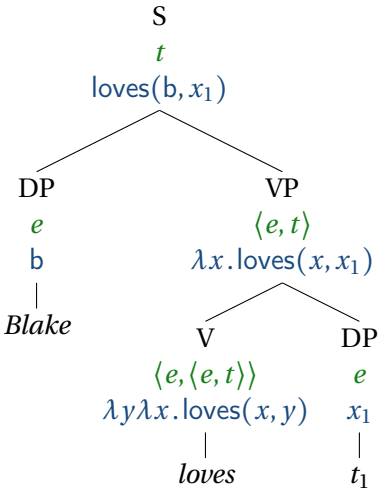
$$he_7 \rightsquigarrow x_7$$

We will see more on the pronoun side of this in Section 7.3.3.[9]

With these assumptions, we derive the representation

$$\mathsf{loves}(\mathsf{b}, x_1)$$

for *Blake loves* t_1:



The translation corresponding to this S node, $\mathsf{loves}(\mathsf{b}, x_1)$, is of type t. Suppose that the complementizer *that* is an identity function of type $\langle t, t\rangle$, so *that* $\rightsquigarrow \lambda p\,.\,p$, where p is a variable of type t. So the relative clause *that Blake loves* t_1 has the same translation, of type t. How does the relative clause end up with a denotation of type $\langle e, t\rangle$? In particular, how do we reach our goal, according to which the relative clause ends up with a translation equivalent to $\lambda x\,.\,\mathsf{loves}(\mathsf{b}, x)$?

[9] The idea of treating traces and pronouns as variables is rather controversial; see Jacobson (1999) and Jacobson (2000) for critique and alternatives.

We can achieve this by assigning the relative clause an interpretation in which a lambda operator binds the variable x_1, thus:

$$\lambda x_1 . \mathsf{loves}(\mathsf{b}, x_1)$$

In principle, the trace might have any index, so we need to know which variable to let the lambda operator bind. We can do this with the help of the index of the relative pronoun. The rule of Predicate Abstraction (also called Functional Abstraction), triggered by the presence of an indexed relative pronoun, turns the appropriate variable from a free one into a bound one. The rule introduces a lambda operator, and so makes use of lambda abstraction in the sense of Chapter 5; but the two should not be confused. Lambda abstraction is an operation of the representation language, available wherever we write a lambda expression, whereas Predicate Abstraction is a rule of semantic composition, triggered by a particular syntactic configuration:

Composition Rule. Predicate Abstraction

If:

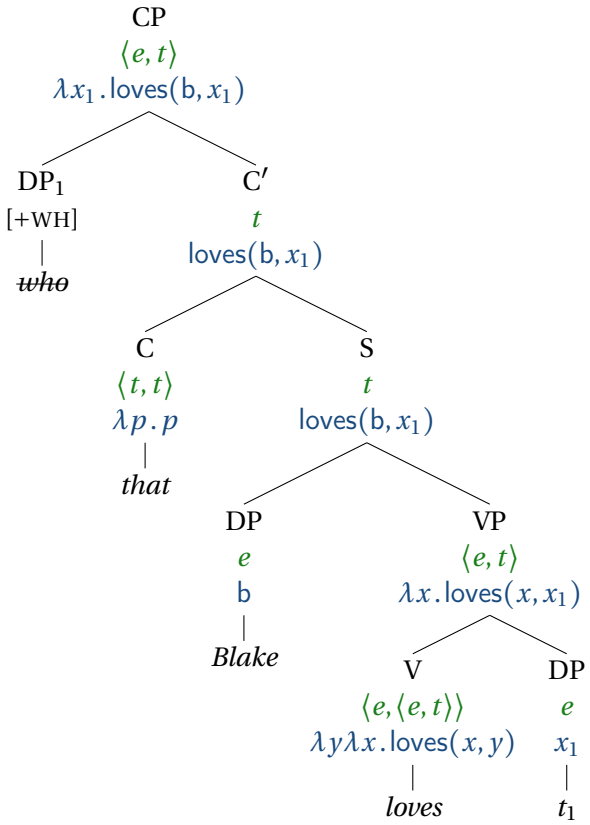
- γ is a syntax tree whose only two subtrees are α_i and β
- α_i is a node carrying the index i and the feature WH
- $\beta \rightsquigarrow \beta'$

Then $\gamma \rightsquigarrow \lambda x_i . \beta'$
where the index on α_i and x_i is the same.

In this rule, the terminal node α_i does not contribute anything other than an index. For this reason, we assume that it carries neither a denotation nor a type, unlike all other terminal nodes.

This rule, Predicate Abstraction, gives us the following analysis of the relative clause, with γ corresponding to the CP node, α_i to the sibling node of C′, and β to the C′ node:

(17)



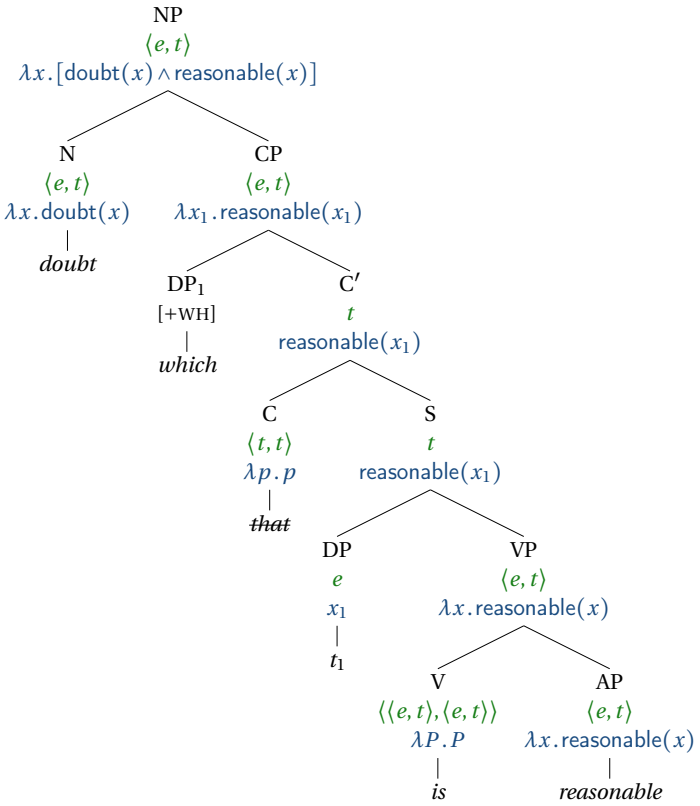
We have reached our goal! The relative clause *that Blake loves* denotes the property of being loved by Blake. Because it translates to an expression of type $\langle e, t \rangle$, it can combine via Predicate Modification with *woman*, giving the property of being in the intersection between the set of women and the set of individuals who Blake loves.

It is important to understand the difference between the denotations of the C′ and CP nodes. The C′ node is of type t, so it denotes a truth value. Whether it denotes True or False depends on what individual the assignment function g assigns to the variable x_1. If that individual is among the individuals loved by Blake, the

node denotes True, otherwise False. The CP node is of type $\langle e, t\rangle$, so it denotes a set of individuals, namely all those individuals that are loved by Blake. Unlike C′, the CP node has a denotation that does not depend on the assignment function g.

Using the same tools, the phrase *doubt which is reasonable* can be given an analysis that accords with Judge Sweeney's intuitions:

(18)



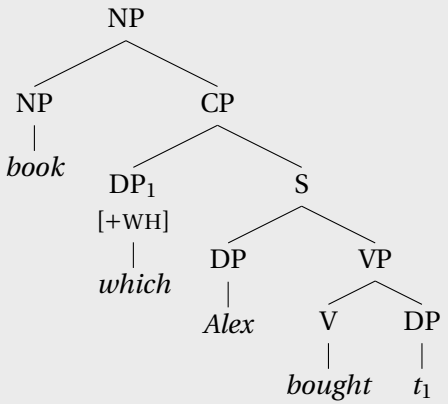
Under this analysis, a doubt which is reasonable is something that is both a doubt and reasonable.

According to the assumptions we have made, a relative pronoun such as *who* or *which* is never assigned a denotation. The

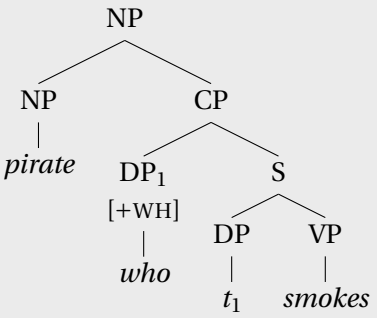
same applies to its silent counterpart in relative clauses that lack overt relative pronouns (whether the complementizer is pronounced, as was illustrated in (17) for *the woman that Blake loves*, or not, as in *the woman Blake loves*). Rather, the contribution of a relative pronoun to the semantic composition of the clause lies in the fact that it triggers the rule of Predicate Abstraction, which gives a denotation for a tree. Thus, relative pronouns don't have a denotation of their own, even though their presence affects the denotation of the constituents that contain them. An expression like this is called SYNCATEGOREMATIC. In contrast, CATEGOREMATIC expressions carry denotations of their own. Most expressions discussed in this book are categorematic.

Exercise 3. Compositionally derive a translation into L_λ for the object relative *book which Alex bought* and the subject relative *pirate who smokes*. LFs are given for you. Assign lexical entries for any undefined terms. The *wh*-words do not need lexical entries; their meaning contribution is syncategorematic.

(i) book which Alex bought



(ii) pirate who smokes

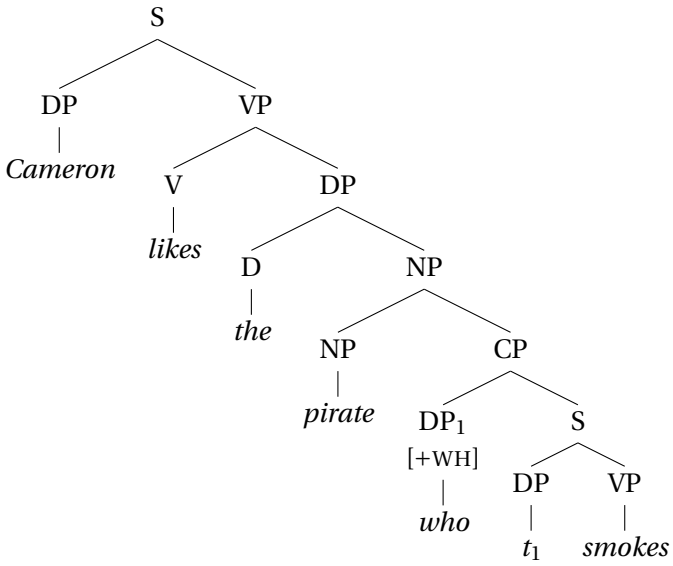


Now let us consider how the NP *pirate who smokes* might be embedded in a larger sentence, like:

(19) Cameron likes the pirate who smokes.

Let us assume that the definite article *the* forms a unit with the NP *pirate who smokes* to form a DP object of the verb *likes* as follows:

(20)

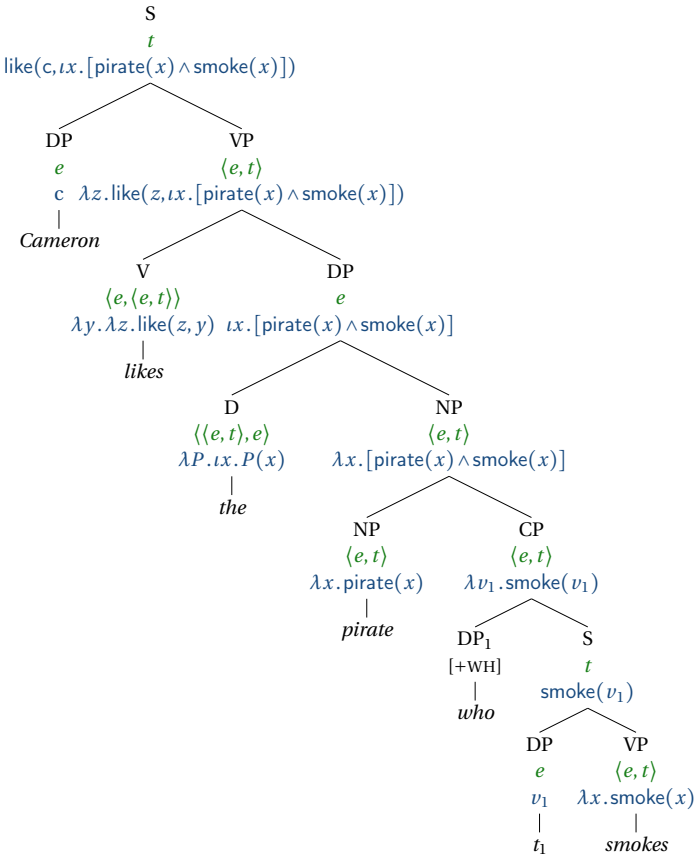


In order to give a compositional analysis of this example, we will need to make an assumption about the denotation of the definite article *the*. Foreshadowing Chapter 8 a bit, let us assume that it is translated as follows:

(21) $\textit{the} \rightsquigarrow \lambda P . \iota x . P(x)$

where P is a predicate (type $\langle e, t \rangle$), and $\iota x . P(x)$, read 'iota x P x' is an expression of type e that denotes the unique satisfier of P (assuming there is one). So the type of the translation for *the* is $\langle \langle e, t \rangle, e \rangle$. We will justify this analysis in greater detail in Chapter 8. With this assumption, we obtain the following translation for (19).

(22)



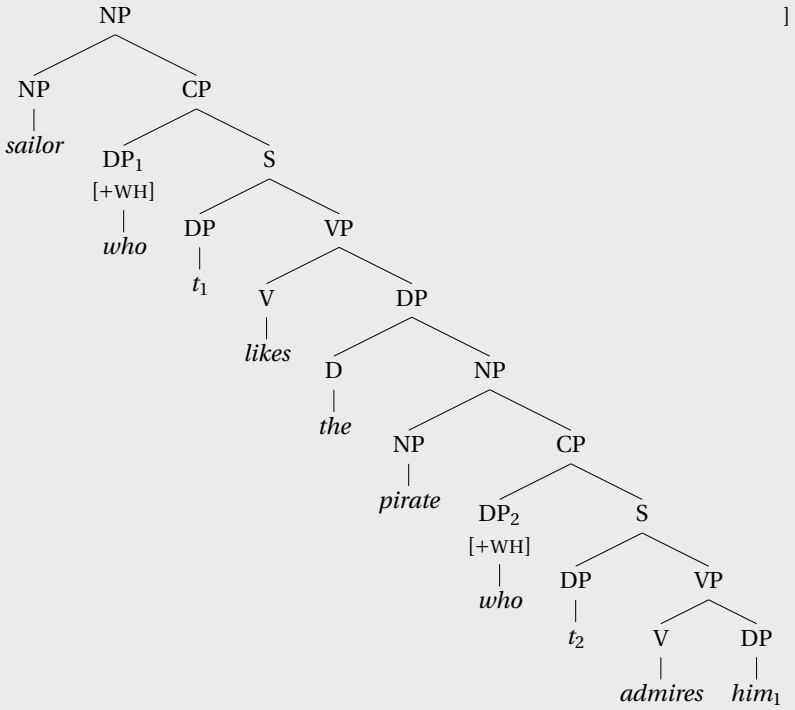
Under the analysis of the sentence provided here, *who smokes* is treated as a RESTRICTIVE RELATIVE CLAUSE, rather than a NON-RESTRICTIVE RELATIVE CLAUSE. In English, a non-restrictive construal of the relative clause is rendered orthographically with the relative clause set off by commas, as in:

(23) Cameron likes the pirate, who smokes.

(This orthographic convention of English can throw off speakers of many European languages, in which restrictive relative clauses

are also set off by commas.) In example (23), with a non-restrictive construal of the relative clause, *the pirate* forms a constituent to the exclusion of the relative clause *who smokes*. Unlike (19), the sentence presupposes that there is one pirate total, and uses the definite description *the pirate* to refer to that individual. The non-restrictive relative clause provides additional information about this individual rather than restricting the set of possible referents for the definite description.

Exercise 4. For each node in the following tree, give the type and a fully beta-reduced translation to L_λ.



7.3.2 Quantifier raising

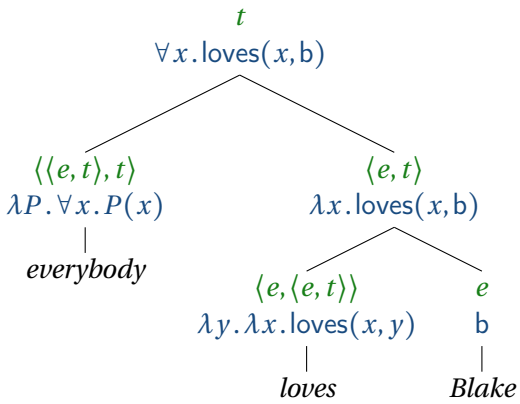
We are now in a position to return to the problem of quantifiers in object position. In some ways, they are entirely parallel to quantifiers in subject position. *Everybody loves Blake* should be translated as:

(24) $\forall x.\mathsf{loves}(x,\mathsf{b})$

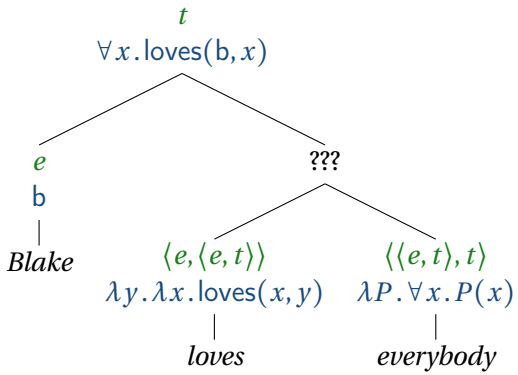
and *Blake loves everybody* should be translated as:

(25) $\forall x.\mathsf{loves}(\mathsf{b},x)$

The first case, with the quantifier in subject position, can be derived compositionally using the tools that we have:



But the case with the quantifier in object position (*Blake loves everybody*) cannot be. Observe what happens when we try:



The transitive verb is expecting an individual, so the quantifier phrase cannot be fed as an argument to the verb. And the quantifier phrase is expecting an $\langle e, t\rangle$-type predicate, so the verb cannot be fed as an argument to the quantifier phrase.

In Chapter 6 we resolved this type mismatch using RAISE-O, a type-shifting operation that lifts the transitive verb to a type that can accept a quantifier phrase as its object. Now that we have introduced the machinery of traces and movement (via Predicate Abstraction), we can develop an alternative solution — one that operates at the level of syntax rather than type-shifting. The two approaches will be compared in Section 7.4.

According to the assumptions we made so far, *everybody* translates as:

(26) $\lambda P.\forall x.P(x)$

As in earlier chapters, this entry glosses over the fact that *everybody* quantifies over animate individuals; the fully specified entry in the toy fragment carries a person restriction. The restriction plays no role in the scope facts at issue here, so we suppress it.

The appropriate value for P here would be a function that holds of an individual if Blake loves that individual:

(27) $\lambda x.\mathsf{loves}(\mathsf{b}, x)$

If we could separate out the quantifier from the rest of the sentence, and let the rest of the sentence denote this function, then we could put the two components together and get the right translation:

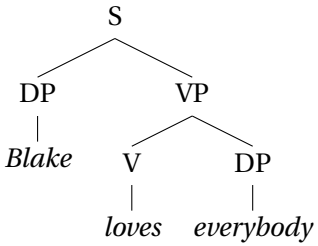
(28) $[\lambda P \forall x.P(x)](\lambda x.\mathsf{loves}(\mathsf{b},x))$

This expression beta-reduces to:

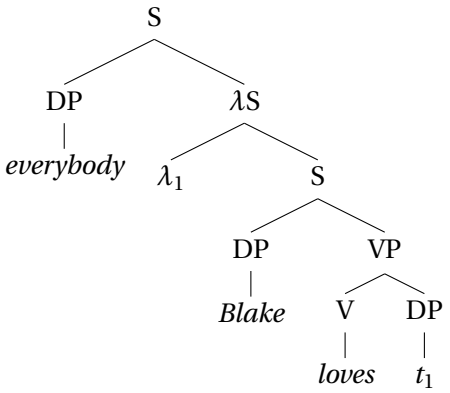
(29) $\forall x.\mathsf{loves}(\mathsf{b},x)$

We can get the components we need to produce the right denotation using the rule of Quantifier Raising. QUANTIFIER RAISING is a syntactic transformation that moves a quantifier (an expression of type $\langle\langle e,t\rangle,t\rangle$) to a position in the tree where it can be interpreted, and leaves a DP trace in its previous position. In terms of 1970s syntax, this transformation occurs not between Deep Structure and Surface Structure, but rather between Surface Structure and another level of representation called Logical Form (LF), as discussed in more detail below. At Logical Form, constituents do not necessarily appear in the position where they are pronounced, but they are in the position where they are to be interpreted by the semantics. Thus the structure in (30a) is converted to the Logical Form representation (30b):

(30) a.



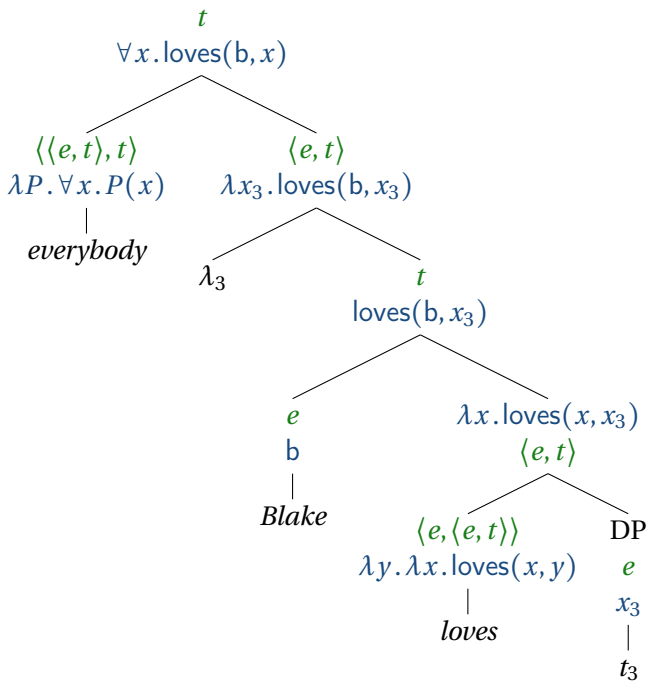
b.



The node with λ_1 in the syntax tree plays the same role as a relative pronoun like *which* in a relative clause: It triggers Predicate Abstraction, which introduces a lambda expression binding the variable corresponding to the trace.

The derivation works as follows. Predicate Abstraction is used at the node we have called λS; Function Application is used at all other branching nodes. The λS node is posited for semantic purposes, and as far as we know, there is no syntactic evidence to support it; it provides a place for the Predicate Abstraction rule to apply. λS was introduced by Heim & Kratzer (1998) and has been widely adopted, though the name we use is specific to our textbook.

(31)

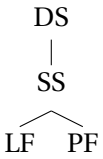


The Quantifier Raising solution to the problem of quantifiers in object position was originally developed in a syntactic theory with several levels of representation:

- Deep Structure (DS): Where active sentences (*John kissed Mary*) look the same as passive sentences (*Mary was kissed by John*), and *wh*- words are in their original positions. For example, *Who did you see?* is *You did see who?* at Deep Structure.
- Surface Structure (SS): Where the order of the words corresponds to what we see or hear (after e.g. passivization or *wh*-movement)
- Phonological Form (PF): Where the words are realized as sounds (after e.g. deletion processes)

- Logical Form (LF): The input to semantic interpretation (after e.g. Quantifier Raising)[10]

Transformations map from DS to SS, and from SS to PF and LF:

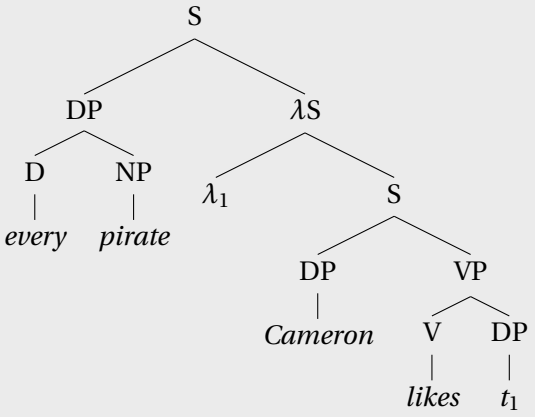


This is the so-called 'T-model', or (inverted) 'Y-model' of Government and Binding theory, motivated originally by Wasow (1972) and Chomsky (1973). Since the transformations from SS to LF happen "after" the order of the words is determined, we do not see the output of these transformations. These movement operations are in this sense COVERT.

Many other transformational generative theories of grammar have been proposed over the years (see Lasnik & Lohndal 2013 for an overview), and many of these are also compatible with the idea of Quantifier Raising; the crucial thing is that there is an interface with semantics (such as LF) at which quantifiers are in the syntactic positions that correspond to their scope, and there is a trace indicating the argument position they correspond to. Quantifier Raising is not an option in non-transformational generative theories of grammar such as Head-Driven Phrase Structure Grammar (Pollard & Sag, 1994) and Lexical-Functional Grammar (Bresnan, 2001); other approaches to quantifier scope are taken in conjunction with those syntactic theories.

[10] 'Logical Form' refers here to a *level of syntactic representation*. A Logical Form is thus a natural language expression, which will be translated into L_λ. It is natural to refer to the L_λ translation as the 'logical form' of a sentence, but this is not what is meant by 'Logical Form' in this context.

Exercise 5. Compositionally derive truth conditions for *Cameron likes every pirate* using Quantifier Raising. You may assume that *every pirate* undergoes Quantifier Raising to produce the following syntax tree at Logical Form:



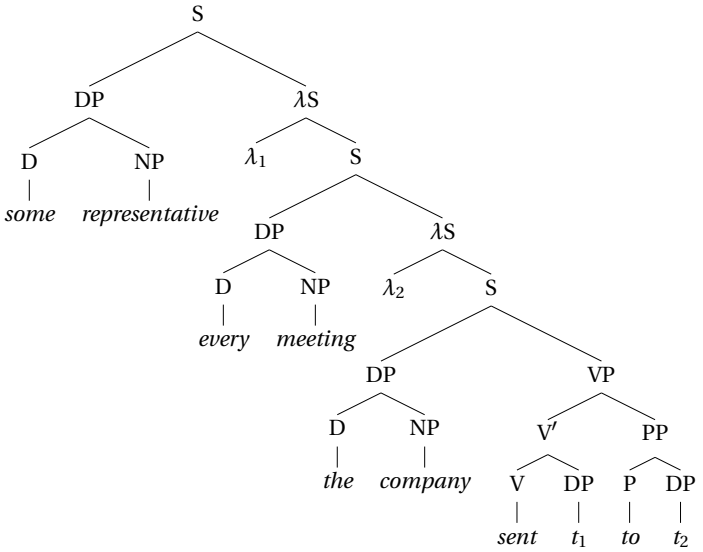
Exercise 6. Produce a translation into the lambda calculus for *Beth speaks a European language*. Start by drawing the LF, assuming that *a European language* undergoes Quantifier Raising. Assume also that the indefinite article *a* can denote what *some* denotes, that *European* and *language* combine via Predicate Modification, and that *speaks* is a transitive verb of type $\langle e, \langle e, t \rangle \rangle$.

There are many factors that play into what sort of scope interpretations hearers are most likely to assign when they hear a sentence, but there can be a preference for surface scope. If both the subject and the object are quantificational, the subject is likely to be interpreted as taking scope over the object. A ditransitive verb like *send*, which takes two objects, evens the syntactic playing field a bit, which can make inverse scope readings more accessible. Consider for example:

(32) The company sent some representative to every meeting.

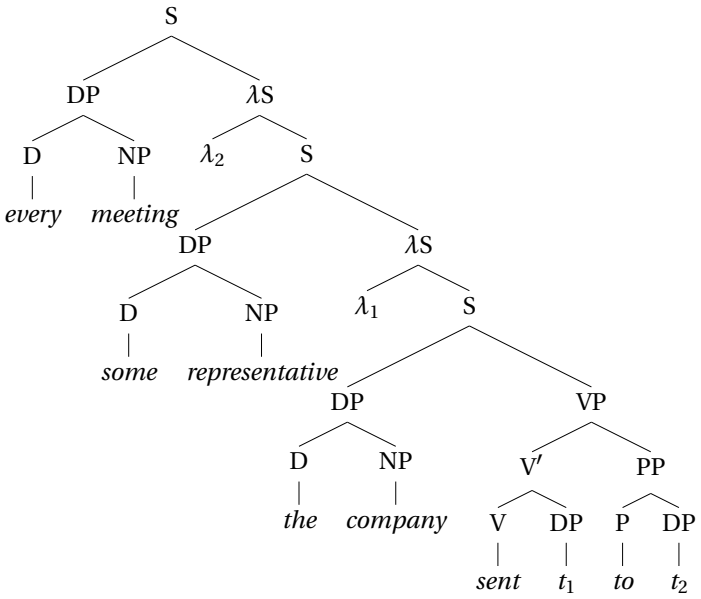
This sentence is scopally ambiguous. We assume that the direct object (DO; *some representative*) is syntactically "higher up in the tree" than the prepositional object (PO; *every meeting*) so that the surface scope reading is the one on which the DO outscopes the PO. Thus on the surface scope reading, *some representative* scopes over *every meeting*. On this reading, the statement is an existential claim about a representative, saying that there is one that the company sent to every meeting.

(33)



On the inverse scope reading, the universal quantifier takes wide scope over the existential quantifier, so the sentence says that every meeting is one that the company sent some representative to.

(34)



Exercise 7. Compositionally derive truth conditions for both the surface scope and the inverse scope readings of *The company sent some representative to every meeting*, using the LFs provided in (33) and (34).

For *send* and *to*, assume the following lexical entries.

- *sent* $\rightsquigarrow \lambda z.\lambda y.\lambda x.\mathsf{send}(x, y, z)$
- *to* $\rightsquigarrow \lambda x.x$

Exercise 8. *Some linguist offended every philosopher* is ambiguous; it can mean either that there was one universally offensive linguist or that for every philosopher there was a linguist, and there may have been different linguists for different philosophers.

Give an LF tree for each of the two readings, and specify the translation into L_λ at every node of your trees.

QR is subject to syntactic constraints. A SCOPE ISLAND is a constituent out of which movement is generally prohibited, including covert movement such as QR. Relative clauses offer a clear illustration: compare (35).

(35) a. John knows a woman from every country.
b. #John knows a woman who is from every country.

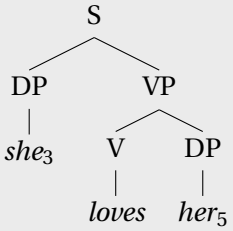
In (35a), *every country* can take wide scope (there is a woman from every country whom John knows). In (35b), however, that reading is absent: the sentence can only mean that John knows a woman who happens to be from every country simultaneously — which is nearly impossible to satisfy.[11] The difference is that in (35b), *every country* is inside a relative clause, and relative clauses are scope islands: QR cannot move *every country* out of them to take scope over the matrix clause. Other widely discussed scope islands include finite embedded clauses (*Blake knows that exactly three people smoke* lacks a reading on which there are exactly three people of whom Blake knows that they smoke) and complex noun phrases more generally. A good theory of quantifier scope must be sufficiently constrained as to rule out scope island violations. If we assume that QR is the same type of movement as *wh*-movement, then we can let the theory of scope islands be inherited from the theory of *wh*-movement (Heim & Kratzer, 1998, 214).

[11] The contrast mirrors a constraint on overt *wh*-extraction: *Which country does John know a woman from?* is grammatical, while **Which country does John know a woman who is from?* is not. This parallelism between extraction and scope is called the EXTRACTION-SCOPE GENERALIZATION.

7.3.3 Pronouns

Recall that the Pronouns and Traces Rule tells us that if α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow x_i$. Thus pronouns and traces are interpreted in the same manner: as variables. In this section we present evidence that pronouns and traces do behave alike, and so should receive a uniform treatment. Whether that treatment should be in terms of variables is a further question, and one on which, as we will see in Section 7.4, not everyone agrees.

Exercise 9. Using the Pronouns and Traces Rule, give translations at every node for the following tree (ignoring the semantic contribution of gender and case):



7.3.3.1 Referential uses of pronouns

Can all pronouns be interpreted as variables? For example, if someone were to point to Cruella De Vil, and say:

(36) She is suspicious.

then this occurrence of *she* would refer to Cruella De Vil. But one could point to Ursula and say the same thing, in which case *she* would refer to Ursula. One doesn't have to point, of course; if Ursula is on TV then she is sufficiently salient for the same utterance to pick her out. Alternatively, one could raise Ursula to salience by talking about her:

(37) Ursula is usually mean, but offered to help Ariel. She is suspicious.

In this case, the pronoun is used ANAPHORICALLY, as it has a linguistic antecedent. In the previous cases, the pronoun is used DEICTICALLY.

In these cases, the pronoun functions as a free variable whose value is supplied by the discourse context — by pointing, visual salience, or prior mention.

7.3.3.2 Pronouns can be bound variables

Not all pronouns are referential, however. There are cases where it is intuitively quite difficult to answer the question "Who/what does the pronoun refer to?":

(38) No woman blamed herself.

(39) Neither man thought he was at fault.

(40) Every boy loves his mother.

It is sometimes said that *No woman* and *herself* are "coreferential" in (38) but this is strictly speaking a misuse of the term "coreferential", because coreference implies reference.

Rather, the pronouns in examples (38)-(40) can be analyzed as bound variables.[12] For example, (38) should be translated as:

(41) $\neg\exists x.[\mathsf{woman}(x) \wedge \mathsf{blamed}(x,x)]$

[12] The terms *free* and *bound* are also used to describe pronouns in BINDING THEORY, the area of syntax that deals with different types of potentially referring expressions like proper names and various types of pronouns. The way that the terms *free* and *bound* are used in that context involves slightly different, albeit related, senses. Here, we use a quite traditional sense of those terms, applying to variables of a formal language that contains variable binders. A variable is free within an expression if it is not bound by any binder within that expression. In syntax, a noun phrase is free in a given expression if it does not have an antecedent within that expression.

This type of evidence supports a treatment of pronouns on which they are translated into the representation language as variables, just like traces.

Another reason to unify the semantics of pronouns and traces is that there are certain cases where pronouns behave almost identically to traces. For instance, regarding the late U.S. Supreme Court Justice Ruth Bader Ginsburg, it was once remarked:

(42) This is an older woman who everyone listens to when **she** speaks.

The alternative with an unpronounced trace (**...who everyone listens to when speaks*) would be ungrammatical; the pronoun *she* rescues the sentence. Pronouns in such configurations are called RESUMPTIVE PRONOUNS. The semantic contribution of a resumptive pronoun is exactly like that of a trace: it is a variable bound by a lambda operator. Thus

who everyone listens to when she speaks

denotes the property of being an x such that everyone listens to x when x speaks. A trace in the same position would contribute identically to the semantics — the difference is syntactic, not semantic.

Another example in which pronouns are interpreted very much like traces is with *such*-relatives, as in:

(43) any book such that Alex bought it

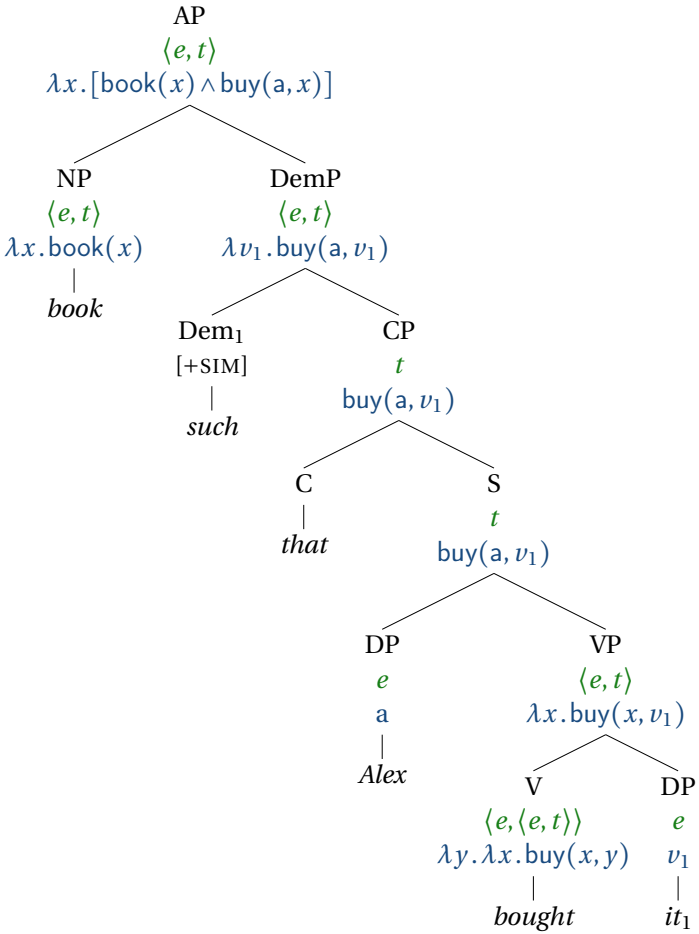
As Quine (1960, 110-11) observes, *such that* divides the two responsibilities of *which*: *it* occupies the singular-term position inside the clause, while *such that* signals the beginning of the relative. Thus *which I bought* becomes *such that I bought it*. English *such* is part of an interesting class of items that go by the name of SIMILATIVES; others include *so* (as in *so tall that he hit his head*) and *as* (as in *as tall as Bill*); see van der Auwera & Sahoo (2020) for a typological perspective on such items. The analysis we give

below is specific to *such*: *so* and *as* in these examples head degree constructions, and we do not assign them the SIM feature or interpret them via Predicate Abstraction.

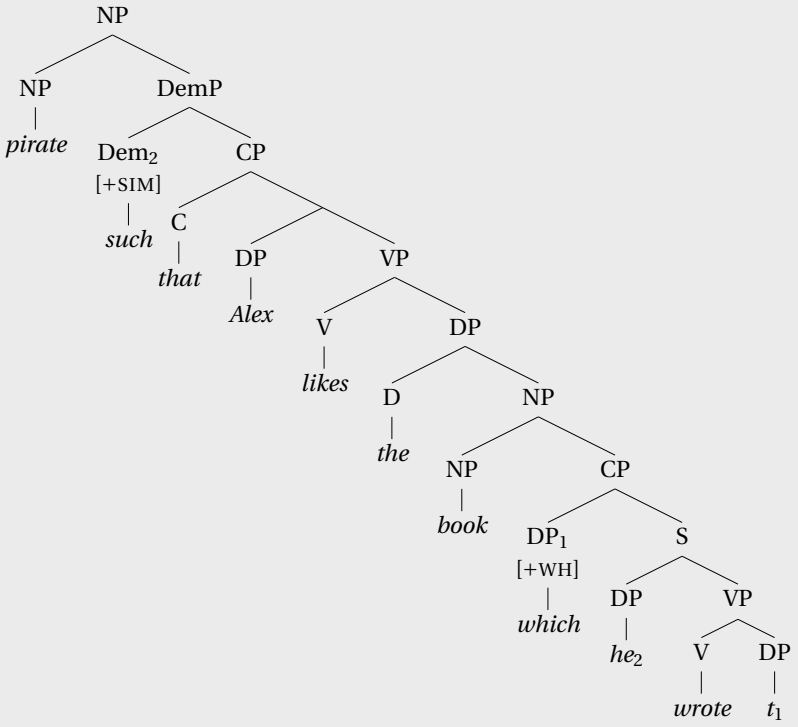
Being exempt from the requirements of *wh-* movement, *such that* relatives have more syntactic flexibility. For example, a *such that* relative can bind into a definite DP that itself contains a relative clause, as in *pirate such that Alex likes the book which he wrote*. Compare *pirate that Alex likes the book which _ wrote*, which is ungrammatical.

Following Heim & Kratzer (1998), let us assume that *such* is coindexed with a pronoun, and that the pronoun is translated as the variable with the corresponding index. We annotate *such* with the feature SIM, short for SIMILATIVE, and treat it as a demonstrative, giving it the category DEM. Its category is not settled in the literature; van der Auwera & Sahoo (2020) survey the cross-linguistic evidence and argue that words of this kind are DEMONSTRATIVE SIMILATIVES, with meanings lying at the intersection of similarity and demonstration. Treating *such* as demonstrative also fits the fact that it is coindexed with a pronoun, which is what licenses the Predicate Abstraction step below. Thus *book such that Alex bought it* will be analyzed as follows:

(44)


AP
⟨e, t⟩
λx.[book(x) ∧ buy(a, x)]
NP
⟨e, t⟩
λx.book(x)
book
DemP
⟨e, t⟩
λv1.buy(a, v1)
Dem1
[+SIM]
such
CP
t
buy(a, v1)
C
that
S
t
buy(a, v1)
DP
e
a
Alex
VP
⟨e, t⟩
λx.buy(x, v1)
V
⟨e, ⟨e, t⟩⟩
λy.λx.buy(x, y)
bought
DP
e
v1
it1

Exercise 10. Compositionally derive a representation in L_λ for *pirate such that Alex likes the book which he wrote*, on a reading where *he* is understood as the pirate.



7.3.3.3 Free vs. bound variable ambiguity

We argued above that pronouns, like logical variables, can be either free or bound. The strict/sloppy ambiguity further illustrates this distinction.

Consider the following exchange from the movie *Ghostbusters*.

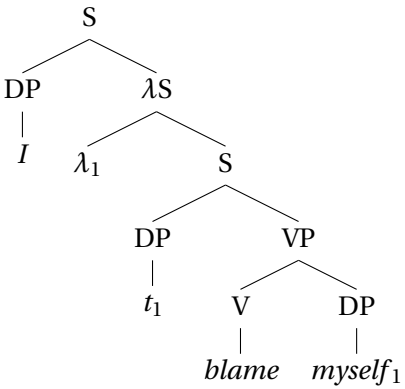
The three Ghostbusters Dr. Peter Venkman, Dr. Raymond Stantz, and Dr. Egon Spengler (played by Bill Murray, Dan Aykroyd, and Harold Ramis, respectively) are in an elevator. They have just started their Ghostbusters business and received their very first call, from a fancy hotel in which a ghost has been making disturbances. They have their proton packs on their backs and they realize that they have never been tested.

(45) Dr. Ray Stantz: You know, it just occurred to me that we really haven't had a successful test of this equipment.
Dr. Egon Spengler: I blame myself.
Dr. Peter Venkman: So do I.

There are two readings of Peter Venkman's quip, a sympathetic reading and a reading on which he is, as usual, being a jerk. On the SLOPPY reading (the sympathetic reading), Peter blames himself. On the STRICT reading (the jerk reading), Peter blames Egon. The strict/sloppy ambiguity exemplified in (45) can be explained by saying that on one reading, we have a bound pronoun, and on another reading, we have a referential pronoun. The anaphor *so* picks up the property 'x blames x' on the sloppy reading.

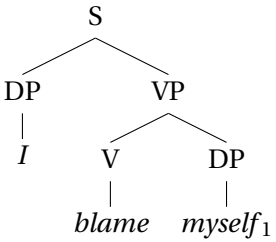
We have not said anything about how to interpret deictic pronouns like *I*, but let us assume that it picks out Peter Venkman just as his name would in the relevant context of utterance. For the reflexive pronoun *myself*, let us assume that it comes with an index that determines which variable it maps to in the representation language, like other pronouns. Suppose that *I* can undergo QR. Then the trace that it leaves behind can be coindexed with the reflexive pronoun, thus:

(46)



The strict reading can be derived from an antecedent without Quantifier Raising:

(47)



We thus capture the strict/sloppy ambiguity by assuming that pronouns are sometimes bound, and sometimes free. This explanation rests on an assumption about how VP ellipsis works: that the anaphor *so* (or a deleted VP) picks up an *LF representation* of the antecedent VP, not merely its surface string or surface semantic value.

Exercise 11. Which reading — strict or sloppy — involves a bound interpretation of the pronoun? Which reading involves a free interpretation?

This way of treating pronouns suggests that assignment func-

tions can be thought of as being provided by the discourse context. A sentence that translates to a logical formula containing a free variable *can* make an interpretable contribution to a discourse, as long as the discourse context provides an assignment function that allows this variable to be interpreted (or a set of such functions constraining the possible values it could take on). Still, it is not appropriate to say *She left!* in a context where your interlocutor has no idea who *she* refers to. This observation could be captured via a requirement that the context specify an interpretation for any free variables that occur in the representation of the meaning of a given text. If *she* translates as x_3, for example, and this variable remains unbound, then the context should determine an assignment function that provides a value for x_3.

We will incorporate presupposition into our theory of semantics. This will be the focus of the next chapter, Chapter 8.

7.4 QR vs. direct compositionality

7.4.1 A type-shifting approach

Quantifier Raising is only one possible solution to the problem of quantifiers in object position. Another approach is to interpret the quantifier phrase *in situ*, i.e., in the position where it is pronounced. In this case one can apply a type-shifting operation to change either the type of the quantifier phrase or the type of the verb. This latter approach, using flexible types for the expressions involved, adheres to the principle of "Direct Compositionality", which rejects the idea that the syntax first builds syntactic structures which are then sent to the semantics for interpretation as a second step. (Direct compositionality is not to be confused with direct interpretation—two totally different ideas.) With direct compositionality, the syntax and the semantics work in tandem, so that the semantics is computed as sentences are built up syntactically, as it were. Jacobson (2012) argues that this is *a pri-*

ori the simplest hypothesis and defends it against putative empirical arguments against it; Bumford & Charlow (2026) develop a directly compositional framework in which functors and monads drive semantic composition.

The phenomena in the following subsections are ones around which the debate between Quantifier Raising and Direct Compositionality has centered. For each, we introduce the phenomenon, show how it can be analyzed using Quantifier Raising and Predicate Abstraction, and indicate where the responses developed within the directly compositional camp can be found. We will not try to settle the dispute here: doing so would mean developing both approaches in comparable detail, and the directly compositional analyses draw on combinatory resources we have not introduced.

Type-shifting rules can target either the quantifier, making it into the sort of thing that could combine with a transitive verb, or the verb, making it into the sort of thing that could combine with a quantifier. On Hendriks's (1993) system, a type $\langle e, \langle e, t\rangle\rangle$ predicate can be converted into one that is expecting a quantifier for its first or second argument, or both.

Another approach uses so-called Cooper Storage, which introduces a storage mechanism into the semantics (Cooper, 1983). This is done in Head-Driven Phrase Structure Grammar (Pollard & Sag, 1994). In brief, the idea is that a syntax node is associated with a set of quantifiers that are "in store". When a node of type t is reached, these quantifiers can be "discharged".

Hendriks defines a general type-shifting schema called ARGUMENT RAISING (not because it involves "raising" of a quantifier phrase to another position in the tree — it doesn't — but because it "raises" the type of one of the arguments of an expression to a more complex type). We will focus on one instantiation of this schema, object raising (RAISE-O), which was introduced in Chapter 6 to handle sentences with quantifiers in object position. We reproduce the definition here for reference. Here and in the following, we will use x for variables associated with the subject, and

y for those associated with the object, wherever possible.

Type-Shifting Rule. Object raising (RAISE-O)
If an English expression α is translated into a logical expression α' of type $\langle e, \langle a, t\rangle\rangle$, for any type a, then α also has a translation of type $\langle\langle\langle e, t\rangle, t\rangle, \langle a, t\rangle\rangle$ of the following form:

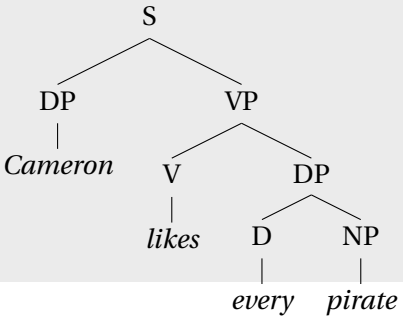
$$\lambda Q_{\langle\langle e,t\rangle,t\rangle} \lambda x_a . Q(\lambda y . \alpha'(y)(x))$$

(unless Q, y or x occurs in α'; in that case, use different variables).

As demonstrated in Chapter 6, applying RAISE-O to a transitive verb allows the object quantifier to remain in situ and combine via Function Application, yielding the expected truth conditions. For example, *Blake loves everybody* receives the analysis $\forall y . \mathsf{loves}(\mathsf{b}, y)$.

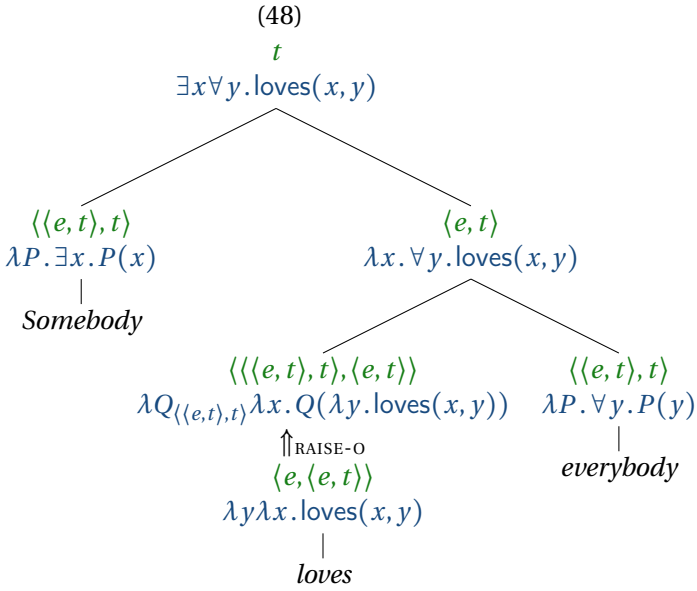
In Exercise 5 you derived truth conditions for *Cameron likes every pirate* using Quantifier Raising. Now try the same sentence using Object Raising, and compare the two analyses.

Exercise 12. Compositionally derive truth conditions for *Cameron likes every pirate* using Object Raising on the transitive verb *likes*. Show both the pre-object-raised and the post-object-raised meaning for the transitive verb on the same node, as in Chapter 6. Compare your result with Exercise 5.



In some situations, it can be useful to apply type-shifting to subject arguments. One such situation stems from scope ambiguities as they occur in sentences with two quantifiers such as *Somebody loves everybody*. Lifting the verb using the Object Raising rule and then combining it with its two arguments results in the surface scope reading, i.e. the reading in which the subject existential takes scope over the object universal. This is shown in the following tree, where subscripts indicate the types of the variables.

(48)



To generate the inverse scope reading, in which the object universal takes scope over the subject existential, we need to lift both arguments of the verb.

Type-Shifting Rule. Subject raising (RAISE-S)

If an English expression α is translated into a logical expression α' of type $\langle a, \langle e, t\rangle\rangle$, for any type a, then α also has a translation of type $\langle a, \langle\langle\langle e, t\rangle, t\rangle, t\rangle\rangle$ of the following form:

$$\lambda y_a \lambda Q_{\langle\langle e,t\rangle,t\rangle} . Q(\lambda x_e . \alpha'(y)(x))$$

(unless Q, x or y occurs in α'; in that case, use different variables).

Applying RAISE-S first, then RAISE-O, produces a doubly-lifted verb that combines with a quantificational subject and a quantificational object. The result for *Somebody loves everybody* is the inverse scope reading $\forall y \exists x . \mathsf{loves}(x, y)$.

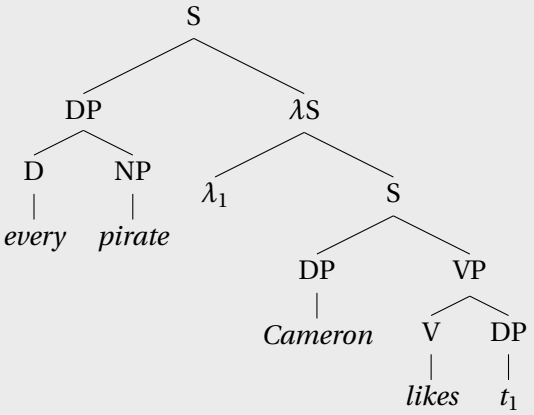
Exercise 13. What happens if we instead apply Object Raising to the verb first, followed by Subject Raising? Draw the resulting derivation tree for *Somebody loves everybody* and state the reading it generates. Can this reading also be generated in a simpler way?

Exercise 14. Quantifiers in object position.

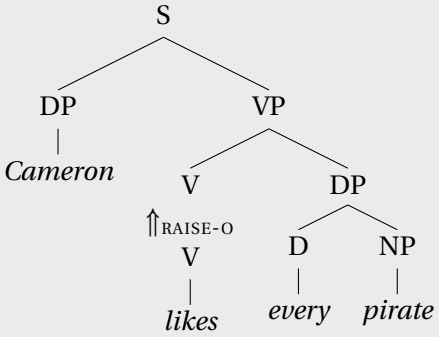
Recall the problem of quantifiers in object position: due to type mismatch, quantifiers cannot be interpreted in object position, unless a type-shifting rule has applied. In order to interpret quantifiers in object position, one of two processes must apply: either Quantifier Raising to elevate the quantificational DP to become the sister of a constituent that includes the subject and verb of the sentence, or type-shifting the verb.

Give two different compositional derivations for the following sentence. For the first one, use Quantifier Raising, and for the second, type-shift the verb.

(i) Cameron likes every pirate



(ii) Cameron likes every pirate



In fact, RAISE-O and RAISE-S are both instances of a more general type-shifting schema due to Hendriks (1993).[13]

[13]The general schema is as follows. If an expression has a translation α' of type $\langle \vec{a}, \langle b, \langle \vec{c}, t \rangle \rangle \rangle$, where $\vec{a}$ and $\vec{c}$ are possibly null sequences of types, then that expression also has a translation of the form $\lambda \vec{x}_{\vec{a}} \lambda Q_{\langle \langle b,t \rangle, t \rangle} \lambda \vec{z}_{\vec{c}} [Q(\lambda y_b [\alpha'(\vec{x})(y)(\vec{z})])]$ (unless x, y, z, or Q occur in α'; in that case, use different variables of the same type).

7.4.2 Inverse linking

One phenomenon that has been presented as an argument for Quantifier Raising is called 'inverse linking', also known as 'binding out of DP' (May, 1985). We give the Quantifier Raising analysis here; for a directly compositional treatment of binding out of DP, including inverse linking, see Barker (2005). A sentence illustrating this phenomenon is in (49).

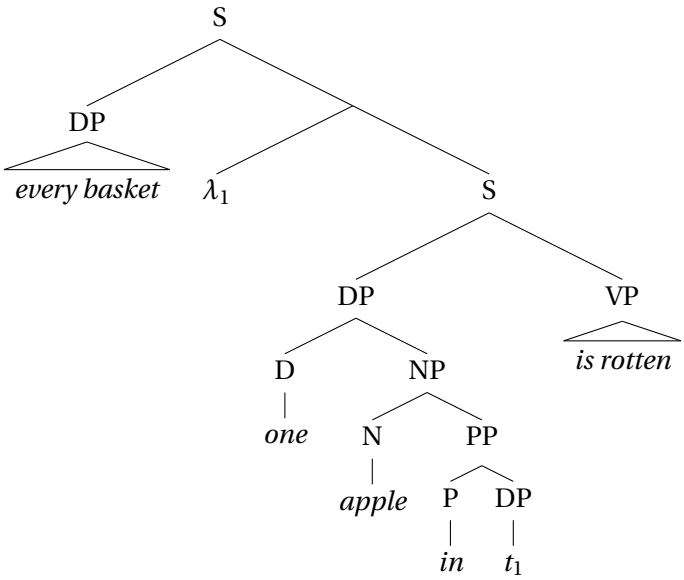
(49) One apple in every basket is rotten.

Assuming that the preposition *in* expects two type e arguments, the quantifier *every basket* cannot be interpreted *in situ* unless *in* undergoes the Object Raising shift. Doing so delivers a reading of this sentence that can be paraphrased: 'One apple that is in every basket is rotten'. But that is not what this sentence intuitively conveys. On the most easily accessible reading of this sentence, the universal quantifier *every basket* takes wide scope over *one apple*; the sentence says that every basket contains one rotten apple.

The QR analysis gives the right reading starting from the following LF:[14]

[14] This LF involves QR of *every basket* out of the subject DP and up to matrix clause level. DPs are generally assumed to be scope islands (see the discussion of scope islands above), which creates a tension: if DP is an island, the movement in (50) should be illicit, just like the wh-extraction in **the basket which one apple in t is rotten* (Heim & Kratzer, 1998, 231). Heim & Kratzer (1998) propose an alternative LF in which *every basket* adjoins only within the subject DP, never leaving it. This DP-internal movement is syntactically licit. To interpret the resulting structure, however, *every basket* must receive a higher-type meaning of type $\langle e, \langle \langle e,t\rangle, t\rangle\rangle \rightarrow \langle \langle e,t\rangle, t\rangle$, rather than the standard $\langle \langle e,t\rangle, t\rangle$. With this flexible-type entry, the complex subject DP denotes a generalized quantifier of type $\langle \langle e,t\rangle, t\rangle$ and combines with *is rotten* by ordinary Function Application, yielding the same truth conditions as (50).

(50)



We treat the cardinal numeral *one* as a uniqueness quantifier, using the ∃! notation to express this.[15]

(51) $one \rightsquigarrow \lambda P.\lambda Q.[\exists!x.[P(x) \wedge Q(x)]]$

Exercise 15. Compositionally derive truth conditions for *One apple in every basket is rotten* based on the LF given in (50) and the lexical entry in (51). Do the truth conditions you derive require that there is an apple that is simultaneously in every basket?

Exercise 16. Give a compositional derivation for the sentence *One apple in every basket is rotten* using the assumption that everything is interpreted *in situ* and the preposition *in* undergoes Ob-

[15] $\exists!x[\phi(x)]$ abbreviates $\exists x[\phi(x) \wedge \forall y[\phi(y) \rightarrow y{=}x]]$: there is an x satisfying ϕ, and any y satisfying ϕ is identical to x.

ject Raising. Use the lexical entry for *one* given in (51). Do the truth conditions you derive require that there is an apple that is simultaneously in every basket?

7.4.3 Antecedent-contained deletion

Another interesting argument that has been made in favor of QR is based on examples like this:

(52) Mary read every novel that John did.

This example involves a kind of ELLIPSIS, also known as DELETION, involving in particular the deletion of a VP. This deletion operation is thought to be licensed under identity with an antecedent VP. Note that describing the phenomenon in these terms already assumes a deletion account of ellipsis, on which the ellipsis site contains unpronounced syntactic structure. That assumption is not neutral in the present debate: directly compositional approaches typically treat VP anaphora without positing deleted structure, and Jacobson (1992) develops a variable-free analysis of antecedent-contained deletion on that basis. For example, in the following case, the deleted VP seems to be 'read *War and Peace*':

(53) I read *War and Peace* before you did ~~read War and Peace~~.

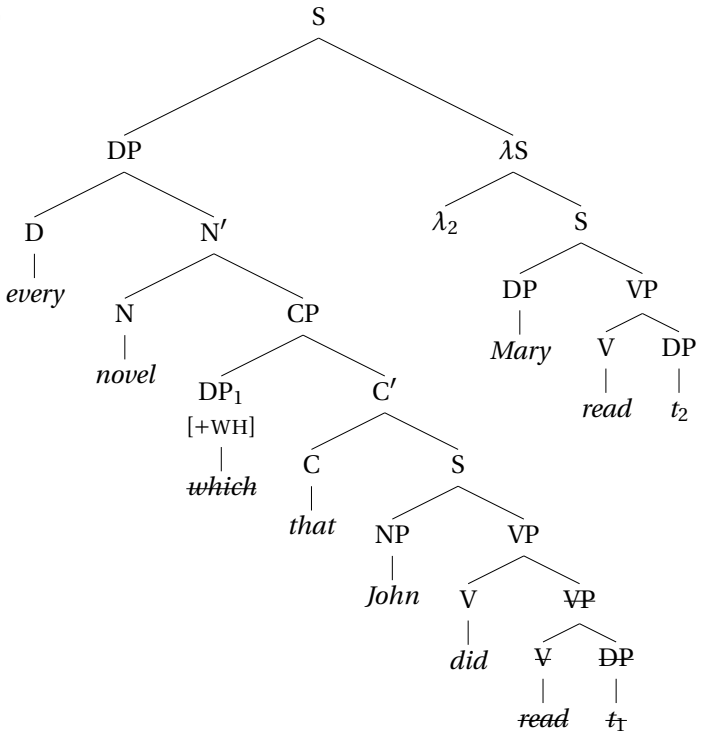
Under a deletion analysis, this VP can be deleted thanks to the fact that it is identical to the antecedent VP in the main clause.

Now, what happens if we try to fill in the antecedent VP in a case like (52)? The antecedent VP *read every novel that John did* contains the ellipsis site itself! We would be in danger of infinite regress if we tried to fill in the antecedent VP: *Mary read every novel that John did read every novel that John did read every novel that...* Since the antecedent VP in this case seems to contain the VP that is deleted, the phenomenon in (52) is known as

ANTECEDENT-CONTAINED DELETION.

Quantifier Raising offers a way out. If the quantifier phrase in (52) undergoes QR, then the antecedent VP no longer contains the deleted VP:

(54)



The antecedent VP (*read* t_2) is now identical to the elided VP (*read* t_1) except for the index on the trace. As long as we assume that two VPs are identical as long as they have the same lexical material and we can ignore the exact indices on the traces, then the ellipsis operation is licensed because the identity criterion is satisfied. Thanks to QR, examples like (52) can be accommodated. A directly compositional alternative that maintains the in situ interpretation of quantifiers has been proposed by Jacobson (1992).

Exercise 17. Give an analysis of (54) with fully beta-reduced λ-expressions at each node.

7.4.4 Reflexive pronouns at a distance

A third phenomenon argued to challenge Direct Compositionality concerns REFLEXIVE PRONOUNS AT A DISTANCE. To set up the challenge, consider the VARIABLE-FREE approach to reflexive pronouns: rather than treating a reflexive like *herself* as a co-indexed variable, one can assign it a lexical entry that reflexivizes any binary relation:

(55) $\textit{herself} \rightsquigarrow \lambda R_{\langle e,\langle e,t\rangle\rangle}.\lambda x.R(x)(x)$

Variable-free semantics and direct compositionality are distinct commitments, and it is worth keeping them apart. Direct compositionality is a claim about architecture: the semantics is computed as the syntax builds, with no separate level of Logical Form. Variable-free semantics is a claim about pronouns and traces: that they are not interpreted as variables. Either can be held without the other. A directly compositional grammar may perfectly well use variables together with type-shifting, and a variable-free treatment of pronouns could in principle be paired with a level of LF. In particular, the entry in (55) is one specific variable-free proposal; it is not something to which direct compositionality is committed, and the difficulties we are about to describe are difficulties for that entry rather than for the architecture.

Applied to a transitive verb, this derives correct truth conditions without any variables. For example, combining (55) with *likes* (type $\langle e,\langle e,t\rangle\rangle$) yields $\lambda x.\mathsf{like}(x,x)$, and a sentence like *every sailor likes herself* receives the right interpretation without QR.

This approach encounters difficulty when the reflexive pronoun is separated from its intended antecedent by intervening

material:

(56) No sailor bought the book about herself.

The subject *no sailor* is the intended antecedent of *herself*, but the reflexive is embedded as the complement of *about*. Under the variable-free approach, *herself* can only interact with its local sister, the preposition *about*: applying (55), the PP *about herself* translates as $\lambda x.\texttt{about}(x,x)$, a predicate true of things that are about themselves. The object DP then denotes the unique book that is about itself. The resulting truth conditions hold that no sailor bought the book about *itself* — the reflexive picks out the book, not any sailor, as its antecedent.

Quantifier Raising avoids this problem: *no sailor* undergoes QR and leaves a trace co-indexed with *herself*. Predicate Abstraction then creates a predicate binding both positions simultaneously, yielding the correct reading.

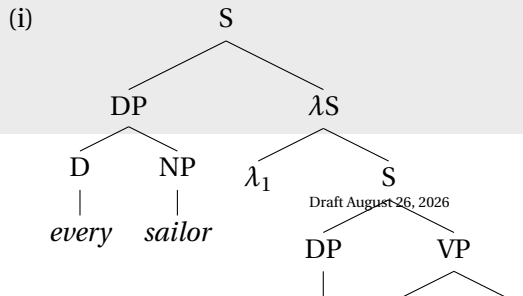
It bears repeating that the difficulty just described is a difficulty for the particular entry in (55), and not for variable-free semantics or direct compositionality in general. Later work has developed directly compositional treatments on which it does not arise; see Bumford & Charlow (2026) in particular.

Exercise 18. Compositionally derive the following sentence, *Every sailor likes herself*, via two different strategies. (i) contains a quantifier-bound pronoun, and (ii) contains no variables.

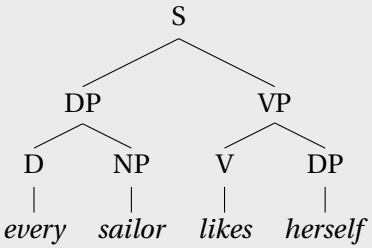
For (i), translate *herself* as in (55). For (ii), provide a denotation for *herself* that will ultimately produce correct truth conditions for *every sailor likes herself* under this parse, where there are no variables in the syntax tree. Ignore the gender component.

Then, calculate the truth conditions for both trees.

(i)

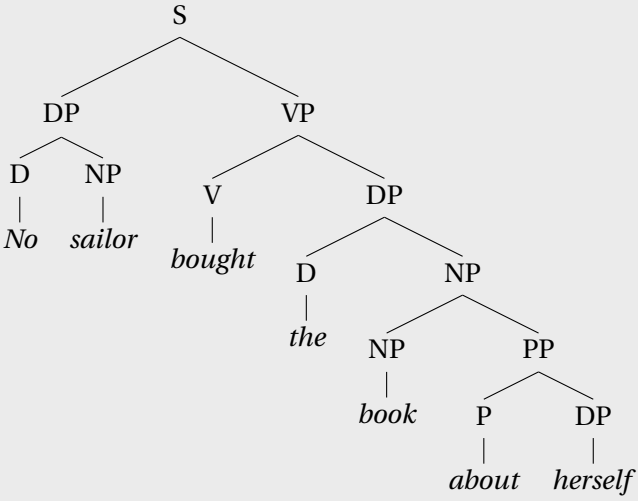


(ii)

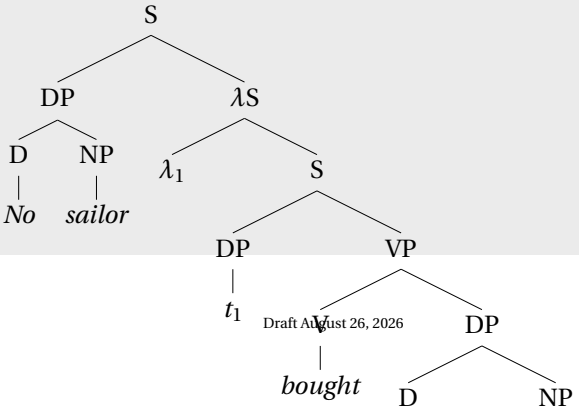


Exercise 19. Two structures for *No sailor bought the book about herself*:

(i)



(ii)



For (i), use the variable-free entry for *herself* from (55). For (ii), use Quantifier Raising. Then calculate the truth conditions for each analysis. Is there something odd about the truth conditions under one of the analyses?

7.4.5 Strict/sloppy ambiguity

The strict/sloppy ambiguity also bears on the debate between Quantifier Raising and Direct Compositionality. Consider the variable-free treatment of reflexive pronouns from (55): assigning *myself* the entry $\lambda R.\lambda x.R(x)(x)$ means that *blame myself* always evaluates to $\lambda x.\mathsf{blame}(x,x)$, independently of the wider syntactic context. This delivers precisely the sloppy reading: when Peter says *so do I* and the VP-anaphor picks up this property as its antecedent, the result is that Peter blames himself.

Deriving the strict reading (Peter blames Egon) requires *myself* to be interpreted as a free variable with Egon as its value, so that *blame myself* denotes the property of blaming Egon. Under the variable-free treatment of reflexive pronouns presented above, where the reflexive pronoun transforms a binary predicate into a reflexive unary one, the strict reading is not derivable.

7.4.6 Summary

We have presented QR-based analyses of four phenomena that have figured prominently in discussions of the relationship between Quantifier Raising and Direct Compositionality: inverse linking, antecedent-contained deletion, reflexive pronouns at a distance, and the strict/sloppy ambiguity in VP ellipsis. In each case, QR provides an elegant solution. Directly compositional alternatives have been developed for all of these phenomena as well (Jacobson, 1992, 1999; Barker, 2005); they do not require a covert level of syntactic representation, but they do require enriching the

inventory of combinatory modes beyond the simple type-shifting approach we have used here. The choice between the two approaches is an active area of research. A meaningful comparison requires specifying the resources available to each framework: as Bumford & Charlow (2026) demonstrate, a sufficiently rich directly compositional architecture can replicate the effects of QR across a wide range of phenomena.

Additions to the toy fragment

The following extends the toy fragment from Chapter 6.

Representation language.

- $\iota x[\phi(x)]$: "the unique x such that $\phi(x)$"; denotes that individual if it exists and is unique, undefined otherwise. We introduce ι here, ahead of its full treatment in Chapter 8, so that the exercises in this chapter can be stated using definite descriptions
- $\exists!x[\phi(x)]$: "there exists exactly one x such that $\phi(x)$"; abbreviates $\exists x[\phi(x) \wedge \forall y[\phi(y) \rightarrow y{=}x]]$

Syntactic transformations.

- **Relative clause formation**: a *wh*-word or relative pronoun moves from its base position, leaving a co-indexed trace; the moved element and its trace share an index
- **Quantifier Raising** (QR): a DP is adjoined to a constituent formed by merging a lambda abstraction node λ_i with a constituent of an appropriate semantic type; a co-indexed DP trace t_i appears in the quantifier's original position; this operation applies at Logical Form (LF)

Composition rules.

- **Predicate Modification** (PM): if $\alpha \rightsquigarrow \alpha'$ and $\beta \rightsquigarrow \beta'$ are both of type $\langle e,t\rangle$, their sister node $\gamma \rightsquigarrow \lambda x.[\alpha'(x) \wedge \beta'(x)]$
- **Predicate Abstraction** (PA): if α_i is an indexed terminal carrying the feature WH, the feature SIM, or is a λ-abstraction index, and $\beta \rightsquigarrow \beta'$, their sister node $\gamma \rightsquigarrow \lambda x_i.\beta'$
- **Pronouns and Traces Rule**: if α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow x_i$

Type-shifting operations.

- **Object Raising** (RAISE-O): generalizes the Chapter 6 version; if $\alpha \rightsquigarrow \alpha'$ of type $\langle e,\langle a,t\rangle\rangle$, then α also has a translation of type $\langle\langle\langle e,t\rangle,t\rangle,\langle a,t\rangle\rangle$:
$\lambda Z\lambda x_a.Z(\lambda y.\alpha'(y)(x))$
- **Subject Raising** (RAISE-S): if $\alpha \rightsquigarrow \alpha'$ of type $\langle a,\langle e,t\rangle\rangle$, then α also has a translation of type $\langle a,\langle\langle\langle e,t\rangle,t\rangle,t\rangle\rangle$:
$\lambda y_a\lambda Z.Z(\lambda x_e.\alpha'(y)(x))$

Lexical entries.

C

- *that* $\rightsquigarrow \lambda p.p$

D

- *the* $\rightsquigarrow \lambda P.\iota x.P(x)$
- *one* $\rightsquigarrow \lambda P.\lambda Q.[\exists!x.[P(x) \wedge Q(x)]]$

Syncategorematically interpreted items.

The following items have no independent lexical denotation; their interpretation is determined entirely by the composition rules above.

D[WH] (relative pronouns): *who, which* — interpreted via Predicate Abstraction

Dem[SIM]: *such* — interpreted via Predicate Abstraction

D (pronouns): *he, she, him, her, it, they, them* — interpreted as x_i via the Pronouns and Traces Rule

8 Presupposition

8.1 Introduction

8.1.1 Back to the square of opposition

There are no dubstep albums by Gottlob Frege (the logician who lived from 1848 to 1925); he just did not make any. So the following sentence is not true:

(1) There are dubstep albums by Frege.

Its negation, naturally, is true:

(2) There are no dubstep albums by Frege.

This is how things usually are; if a sentence is not true, then its negation is true. But this is not always the case.

The following sentence, in which *every* combines with *dubstep albums by Frege*, is not felt to be true:

(3) Every dubstep album by Frege is famous.

Yet few would assent to its negation:

(4) Not every dubstep album by Frege is famous.

Thus neither the original sentence nor its negation is felt to be true. How can this be?

One hypothesis is that (3) is neither true nor false: it has a

PRESUPPOSITION FAILURE, because its presupposed content—that there exist dubstep albums by Frege—is false. Under this hypothesis, assuming that the negation of an undefined sentence is also undefined, (4) would be undefined as well, explaining why neither sentence feels straightforwardly true or false.

More generally, we might hypothesize that a sentence of the form *Every S is P* receives the truth value # whenever there are no Ss. We did not address this sort of empty-restrictor case in Chapter 1. Table 8.1 illustrates what the hypothesis predicts for the sailor/pirate scenario, covering all eight combinations of empty and non-empty restrictor and nuclear-scope sets.

This hypothesis is grounded in the *semantic definition* of presupposition: A presupposes B when B must be true for A to have any truth value at all. Under this definition, the determiner *every* carries a PRESUPPOSITION of existence: *Every S is P* presupposes that there is at least one S.

Let us consider what this hypothesis means for the semantic relations binding together the four corners of the square of opposition.

The definitions from Chapter 1 extend to the three-valued setting without modification, because they are stated in terms of T and F and # is neither. **Entailment** requires only that whenever A is T, B is T too; cases where A is # place no condition on B, so the empty-restrictor cases (vi, viii), where both $\boxed{\text{A}}$ and $\boxed{\text{I}}$ are #, are simply cases where the antecedent of the condition is not met. For **contradictory** pairs ($\boxed{\text{A}}$/$\boxed{\text{O}}$ and $\boxed{\text{E}}$/$\boxed{\text{I}}$), the definition requires that they cannot both be T and cannot both be F. In empty-restrictor cases both members of the pair are #—neither T nor F—so neither clause is violated; # cases are a new third possibility that the two-valued definition simply did not need to anticipate. **Contrary** ($\boxed{\text{A}}$/$\boxed{\text{E}}$) and **subcontrary** ($\boxed{\text{I}}$/$\boxed{\text{O}}$) pairs work analogously: contraries cannot both be T but can both fail to be T (in the three-valued setting, by being # rather than F); subcontraries cannot both be F but can both fail to be F (again, by both being #).

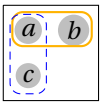
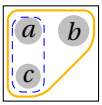
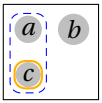
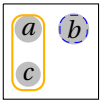
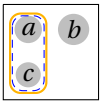
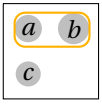
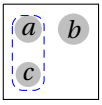
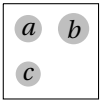
			A *Every S is P*
i.		sailors: a, c; pirates: a, b	F
ii.		sailors: a, c; pirates: a, b, c	T
iii.		sailors: a, c; pirates: c	F
iv.		sailors: b; pirates: a, c	F
v.		sailors: a, c; pirates: a, c	T
vi.		no sailors	#
vii.		no pirates	F
viii.		no sailors or pirates	#

Table 8.1: Truth values for A (*Every S is P*) across eight cases. S = *sailor*, P = *pirate*. Dashed outline = sailors; solid outline = pirates.

With the definitions of Chapter 1 applied without modification, then, all four relations of the medieval square of opposition are satisfied under the presuppositional analysis, as shown in Figure 8.1.[1] The result is not an accident. The models divide into two kinds: where the restrictor is non-empty, nothing is #, the logic behaves just as the classical two-valued logic does, and the square is valid for the reasons the Aristotelian tradition already recognized; where the restrictor is empty, both members of every pair are #, and since each relation in the square is stated as a requirement that certain combinations of T and F not arise, every such requirement is met vacuously. The square is preserved because one kind of model validates it classically and the other validates it vacuously.

Exercise 1. The closing discussion above hints that applying the presuppositional treatment "uniformly to all four corners" of the square is what restores the medieval relations. This exercise asks you to work out what that means for the E corner, *No S is P*.

(a) Suppose *no* does *not* presuppose the existence of sailors, so that *No sailor is a pirate* receives the truth value T in case vi (no sailors at all). What value does O (*At least one sailor is not a pirate*) receive in that case, and what does this mean for the entailment from E to O that the medieval square requires?

(b) Propose a lexical entry for *no* that presupposes the existence of at least one entity satisfying the restrictor, analogous to the

[1] The classical alternatives for the empty-restrictor case give different results. Treating *Every S is P* as vacuously T when there are no Ss—as in standard predicate logic, where the universal quantifier is defined purely as a subset condition—severs the A⇒I entailment: A would be T yet I F in the empty-restrictor cases. Treating it as F instead restores that entailment but breaks the A/O contradictory relation: with no sailors, O (*At least one sailor is not a pirate*) is also F, since there are no sailors to be non-pirates, leaving both F simultaneously.

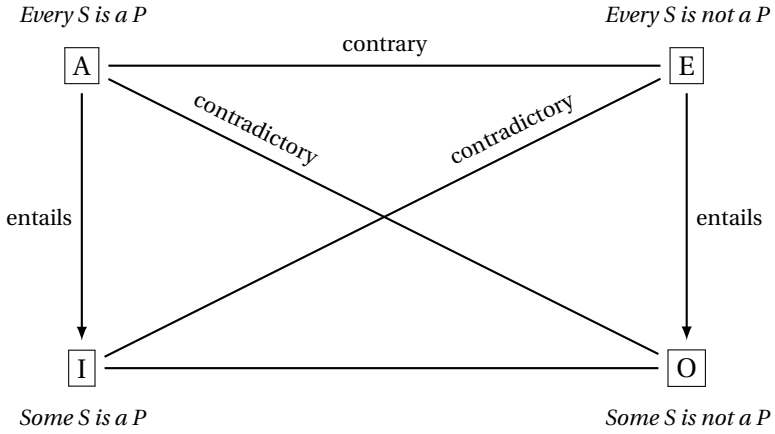


Figure 8.1: The square of opposition, as envisaged by medieval scholars. Shorthand: *S* = *sailor*; *P* = *pirate*.

entry for *every* in (21).

(c) Using your entry, give the truth values of all four corners in case vi, and verify that the relations in Figure 8.1 hold.

8.1.2 Diagnosing presuppositions

The previous section motivated a semantic view of presupposition: A presupposes B if and only if B must be true in order for A to have an ordinary truth value. There is also a pragmatic way of thinking about presupposition, one which leads naturally to certain diagnostic tests for it. What a *speaker* presupposes is what they take for granted, treating it as uncontroversial and known to everyone participating in the conversation (Stalnaker, 1978). The idea of a *sentence* presupposing something follows: a sentence A presupposes B if uttering A in any given context signals that the

speaker takes B for granted.

As Chierchia & McConnell-Ginet (2000, 28) write, "If A presupposes B, then A not only implies B but also implies that the truth of B is somehow taken for granted, treated as uncontroversial." Furthermore,

> If A presupposes B, then to assert A, deny A, wonder whether A, or suppose A – to express any of these attitudes toward A is generally to imply B, to suggest that B is true and, moreover, uncontroversially so. That is, considering A from almost any standpoint seems already to assume or presuppose the truth of B; B is part of the background against [which] we (typically) consider A.

Thus if A presupposes B, then the negation of A, a sentence with *maybe* prepended to A, and a conditional with A as antecedent will all also presuppose B. The following sentences, for instance, all imply that Frege made at least one dubstep album:

(5) Not every dubstep album by Frege is famous.

(6) Maybe every dubstep album by Frege is famous.

(7) If every dubstep album by Frege is famous, they are all on Spotify.

Every one of these sentences shares the implication; this is characteristic of presupposition. In general, sentences that embed the original sentence under negation, questions, modals, and conditionals are used to test for presuppositions. This is called the FAMILY-OF-SENTENCES TEST.[2]

The part of the sentence (word or construction) that carries this signal that something is being presupposed is called a PRE-

[2] Strictly speaking, questions do not entail their presuppositions in the usual sense, since questions do not have truth values. The family-of-sentences test can be extended to cover questions by requiring that the presupposition be implied regardless of the speech act — assertion, denial, or inquiry.

SUPPOSITION TRIGGER. The word *every* triggers a presupposition of existence. Another example of a presupposition trigger is the adverb *still*, in the sense "up to the present". For example, if I said (8), I would signal (9) through a presupposition.

(8) Natalie Portman still speaks French.

(9) Natalie Portman spoke French in the past.

Although it is not an *ordinary* entailment, the relation between these sentences *is* arguably some form of entailment; in every situation where (8) is true, (9) is also true. This can also be shown using the defeasibility test. But presuppositions differ from ordinary entailments, as you can see from what happens when they are negated. Suppose we negate (8) as follows:

(10) It's not the case that Natalie Portman still speaks French.

This sentence denies that Natalie Portman currently speaks French but still implies that she spoke French in the past.

In fact, merely *supposing* that Natalie Portman still speaks French also yields the implication that she spoke French in the past.

(11) If Natalie Portman still speaks French, then she might enjoy this poem.

Here we have placed *Natalie Portman still speaks French* in the ANTECEDENT position (the 'if' part) of a CONDITIONAL statement. (The 'then' part is called its CONSEQUENT.) Normally, material that is in the antecedent of a conditional is not implied. For example, the following sentence does not imply that Natalie Portman speaks French:

(12) If Natalie Portman speaks French, then she might enjoy this poem.

The antecedent of a conditional is for ideas that are merely entertained for the purpose of exploring a hypothetical possibility; the

speaker normally does not commit herself to the material here. But presupposed information still 'pops out' from the antecedent of a conditional, as it were. In other words, the presupposition PROJECTS from the antecedent of the conditional (and from under negation).

We see this with questions as well. If someone were to ask,

(13) Does Natalie Portman speak French?

they would not be implying that Natalie Portman spoke French, of course. And yet:

(14) Does Natalie Portman still speak French?

does imply that Natalie Portman spoke French at some time in the past. The presupposition projects out of the yes/no question.

In general, presuppositions can be distinguished from entailments using this PROJECTION TEST, which assesses whether the inference in question 'projects' over negation, from the antecedent of a conditional statement or over question-formation. What these environments have in common is that they are ENTAILMENT-CANCELING environments; environments where entailments normally go to die. But presuppositions thrive in these environments. To test whether an inference from A to B is an ordinary entailment or a presupposition, one embeds A in an entailment-canceling environment, and observes whether the B sentence is still implied. If so, then the inference projects, and is therefore behaving as a presupposition.

So far we have mentioned two presupposition triggers: the quantificational determiner *every* and the adverb *still*. Other presupposition triggers include the quantificational determiners *both* and *neither*, factive adjectives (e.g., *glad, annoying*), factive verbs (e.g., *know, remember, realize*), possessives, exclusives (e.g., *only, merely, sole*), and the definite determiner *the* (also called a definite article). Here are some examples (where ≫ signifies 'presupposes'):

(15) a. Neither candidate is qualified.
≫ There are exactly two candidates.
b. Ed is glad we won.
≫ We won.
c. Ed knows we won.
≫ We won.
d. Ed's son is bald.
≫ Ed has a son.
e. Only Ed came.
≫ Ed came.
f. The balcony is lovely.
≫ There is exactly one relevant balcony.

Let us consider the case of possessives. Example (16a) implies (16b), broadly speaking; anyone who heard (16a) would certainly conclude that (16b) is true, assuming they trusted the speaker.

(16) a. Kim's twin sister lives in Austin.
b. Kim has a twin sister.

Does this implication project? Let us apply the projection test. To do so, we'll need to embed (16a) in an entailment-cancelling environment, such as negation, the antecedent of a conditional, or a *maybe* statement. Let's try all three, just to be on the safe side:

(17) a. **Negation**
Kim's twin sister doesn't live in Austin.
b. **Antecedent of a conditional**
If Kim's twin sister lives in Austin, then Kim has probably eaten at Torchy's Tacos.
c. ***Maybe***
Maybe Kim's twin sister lives in Austin.

These sentences all imply that Kim has a twin sister. So the inference projects.

Exercise 2. Use the projection test to determine whether the following implications are entailments or presuppositions. Explain how the test supports your conclusion.

(a) Frege came again.
Frege has come sometime in the past.

(b) Frege came yesterday.
Frege has come sometime in the past.

Exercise 3. Consider the following two sentences.

(18) a. John succeeded in learning to play the guitar.
b. John failed at learning to play the guitar.

Intuitively, both sentences imply that John tried to learn to play the guitar (19a), but the *succeed* sentence implies that he did (19b), and the *fail* sentence implies that he did not (19c).

(19) a. John tried to learn to play the guitar.
b. John learned to play the guitar.
c. John didn't learn to play the guitar.

So there are four implications under consideration:

(20) a. (18a) 'succeed' → (19a) 'try';
b. (18b) 'fail' → (19a) 'try';
c. (18a) 'succeed' → (19b) 'did';
d. (18b) 'fail' → (19c) 'didn't'.

For each of these in turn, determine whether it is an implicature, an ordinary entailment, or a presupposition. First, determine whether it is an implicature or an entailment (ordinary or presupposition) using the defeasibility and reinforcement tests, and

then, if it is an entailment, determine whether it is an ordinary entailment or a presupposition using projection from negation, the antecedent of a conditional, and a yes/no question.

Be sure to include all of the relevant examples, observations, and reasoning in your answer, and summarize your findings by saying in general what is entailed, presupposed, and implicated (if anything), by a sentence of the form *X succeeded in Y*, and do the same for *X failed at Y*.

The formalization of presupposition requires extending our logic beyond two truth values. The next section develops that extension, beginning with a return to the square of opposition.

8.2 Presupposition and three-valued logic

With the motivation for a third truth value established, we now develop the formal system that implements it.

8.2.1 Designing a three-valued logic

We treat presupposition failures as cases where an expression lacks a classical truth value. We introduce a third truth value, #, to use as the denotation for sentences containing presupposition failures. Under Weak Kleene, # can be thought of as 'nonsense'—the sentence has gone awry—or as 'void': a sentence that simply lacks a proper truth value. We write $\#$ (without subscript) as shorthand for $\#_t$ throughout; subscripted variants for other types are introduced when the full type-theoretic framework is developed in Section 8.3.4.

Now, in setting up a logic with three truth values, a number of decisions have to be made. For example, what if ϕ is undefined and ψ is true—is $[\phi \wedge \psi]$ undefined or false? If we take undefinedness to represent 'nonsense', then presumably the conjunction of

$\wedge$	T	F	#
T	T	F	#
F	F	F	#
#	#	#	#

$\vee$	T	F	#
T	T	T	#
F	T	F	#
#	#	#	#

	$\neg$
T	F
F	T
#	#

Table 8.2: Truth tables for the Weak Kleene connectives

nonsense with anything is also nonsense. The same applies for disjunction, and the negation of an undefined formula is also presumably undefined. This perspective leads to the truth tables in Table 8.2. In the truth tables for the binary connectives, the truth value of one conjunct (or disjunct) is represented by the row labels, and the truth value of the other is represented by the column labels. The tables are symmetric, so it doesn't matter which is which. The value in the table is the value for the conjoined (or disjoined) formula. These connectives are called the WEAK KLEENE connectives, after the American mathematician Stephen Cole Kleene.[3]

8.2.2 Definedness conditions

The PARTIAL OPERATOR (∂) is a general formal tool for representing definedness conditions. It will be the main device for captur-

[3] An alternative set of connectives, the STRONG KLEENE connectives, treats the third truth value as 'unknown' rather than 'nonsense'. Under Strong Kleene, the conjunction of an unknown value with F is F, and the disjunction of an unknown value with T is T. For example, *The king of France is wise and France is located in Africa* comes out as # under Weak Kleene but F under Strong Kleene, and *The king of France is wise or France is located in Europe* comes out as # under Weak Kleene but T under Strong Kleene. Without any additional mechanisms, the Strong Kleene connectives give us more flexibility: Weak Kleene connectives can be defined in terms of Strong Kleene ones, but not vice versa. Furthermore, the Weak Kleene connectives predict that presuppositions always project, while the Strong Kleene connectives allow some presuppositions to be cancelled (see Section 8.5 for further discussion).

ing presuppositions in this chapter. In Section 8.3, we will see that some presuppositions—those involving definite descriptions and possessives—arise from an independent source: a presupposition of existence and uniqueness encoded in the ι operator. The partial operator is more general, and can be used to represent the presuppositions of a wide range of expressions.

The ∂ operator comes from Beaver & Krahmer (2001) and is pronounced 'presupposing that'. It is of type $\langle t, t\rangle$: it maps a formula to another formula. For example,

$$\partial(\exists x.P(x))$$

can be read as 'presupposing that there is at least one P'.

As a first application, we can give a formal account of the existence presupposition of *every* that motivated our study of three-valued logic:

(21) $\quad \textit{every} \rightsquigarrow \lambda P \lambda Q.[\partial(\exists x.P(x)) \wedge \forall x.[P(x) \rightarrow Q(x)]]$

This treatment gives rise to an undefined value for *Every dubstep album by Frege* in models where there are no dubstep albums by Frege (such as the one corresponding to reality), capturing the intuition that the sentence is neither true nor false. The prediction depends on conjunction being Weak Kleene: because the first conjunct is undefined, the conjunction as a whole is undefined regardless of the value of the second conjunct.

In order to be able to give translations like (21), we need to augment L_λ to handle formulas containing the ∂ symbol. Let us call our new language L_∂. In this new language, $\partial(\phi)$ will be a kind of expression of type t. Its value will be 'true' if ϕ is true and 'undefined' otherwise. While the logics in previous chapters were classical and therefore two-valued, this new language is a THREE-VALUED LOGIC: a logic in which there are three truth values to be assigned to sentences. To implement this, we add the following rules:

Syntax Rule: Definedness conditions
If ϕ is an expression of type t, then $\partial(\phi)$ is an expression of type t.

Semantic Rule: Definedness conditions
If ϕ is an expression of type t, then:

$$[\![\partial(\phi)]\!]^{M,g} = \begin{cases} \top & \text{if } [\![\phi]\!]^{M,g} = \top \\ \# & \text{otherwise.} \end{cases}$$

Not all presuppositions involve the presupposition of existence for a given referent. The determiners *both* and *neither*, for example, come with presuppositions; accordingly, they are called PRESUPPOSITIONAL DETERMINERS. In a context where three candidates are applying for a job, it would be quite odd for someone to say either of the following:

(22) a. Both candidates are qualified.
b. Neither candidate is qualified.

If there were only two candidates and both were qualified, then (22a) would clearly be true and (22b) would clearly be false. But with any number of candidates other than two, it is odd to say that these sentences are true. Applying the projection test, we can see that this inference survives embedding under entailment-cancelling operators:

(23) a. It's not true that both candidates are qualified.
b. It's not true that neither candidate is qualified.

(24) a. If both candidates are qualified, then we will have a round of interviews.
b. If neither candidate is qualified, then we will have to expand the search.

(25) a. Maybe both candidates are qualified.
b. Maybe neither candidate is qualified.

All of these imply that there are two candidates. These results support the idea that *both candidates* and *neither candidate* come with a presupposition that there are exactly two candidates.

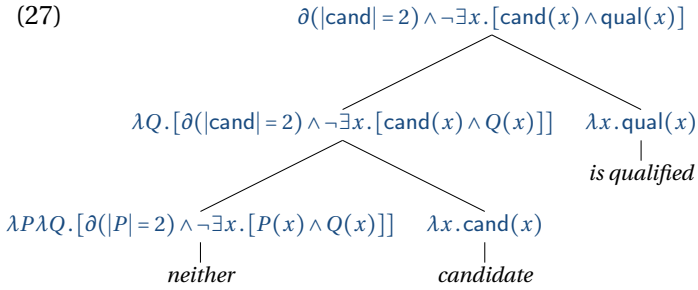
We will set aside *both* and focus on *neither*, in its use as a determiner as in (22b). We can model its presupposition by treating *neither* as a variant of *no* that is only defined when its argument is a predicate with exactly two satisfiers. Let us use $|P| = 2$ ('the cardinality of P is 2') as a shorthand way of expressing the fact that predicate P has exactly two satisfiers.[4] Using the ∂ operator, the lexical entry for *neither* is:

(26) $\textit{neither} \rightsquigarrow \lambda P \lambda Q . [\partial(|P| = 2) \wedge \neg \exists x . [P(x) \wedge Q(x)]]$

This says that *neither* is basically a synonym of *no*, carrying an extra presupposition: that there are exactly two Ps.

We can now work out the full analysis for (22b). The lexical entry gives us the following derivation, where beta-reduced variants of the translations are given at each node:

(27)



The translation for the whole sentence should have a defined value in a model if $|\mathsf{cand}| = 2$ is true in the model. If it has a defined value, then its value is equal to that of $\neg \exists x . [\mathsf{cand}(x) \wedge \mathsf{qual}(x)]$.

[4] $|P| = 2$ is short for $\exists x \exists y [\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg \exists z [\neg(z = x) \wedge \neg(z = y) \wedge P(z)]]$.

Exercise 4. (i) Give a derivation tree for the following sentences, specifying a translation at each node:

(a) *Neither elevator is new.*

(b) *Every elevator is new.*

(c) *Not every elevator is new.*

Assume that *new* denotes a predicate of individuals. You can use the following translation for *not* in the third sentence:

$$\lambda Q_{\langle\langle e,t\rangle,t\rangle}\lambda P_{\langle e,t\rangle}.\neg Q(P)$$

(ii) Based on the truth conditions you've derived for the sentences in the first part, which of those sentences are predicted to presuppose that there are elevators (if any)?

Exercise 5. Gender on Adjectives in Russian.

In Russian, the name Sasha can be used for either male or female. If you say *Sasha uchitel*, that means *Sasha is a teacher*, and implies that Sasha is a *man*.

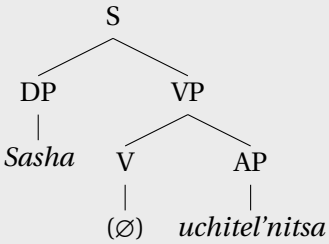
However, if you say *Sasha uchitel'nitsa*, then it means *Sasha is a teacher*, and imples that Sasha is *female*. Negating these sentences (producing *Sasha ne uchitel* and *Sasha uchitel'nitsa*, respectively), does not affect the gender implications; the masculine adjective conveys that the subject is male, and mutatis mutandis for the feminine adjective. This suggests that the inference is a presupposition.

Give a lexical entry for *uchitel'nitsa* on which the gender component is presupposed, so the following sentence entails that Sasha is a teacher and presupposes that Sasha is female (also give a lexical entry for *Sasha*). Then, compositionally derive the entire

tree. As above, use δ to represent the partial operator (cf. the textbook). Hint: look at the lexical entry for *neither* for inspiration.

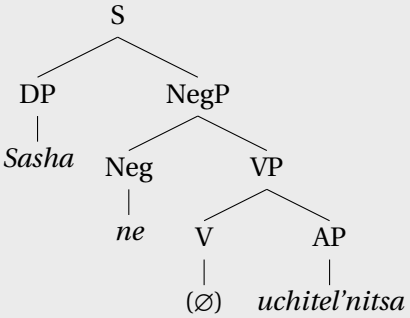
(i) Sasha uchitel'nitsa (Sasha is a teacher, implying that Sasha is female).

- *uchitel'nitsa* $\rightsquigarrow$?
- *Sasha* $\rightsquigarrow$?



Next, using the same lexical entries, give a compositional analysis for the negated sentence '*Sasha ne uchitel'nitsa*. This sentence should deny that Sasha is a teacher, and still presuppose that Sasha is female. Hint: use prior homeworks for a lexical entry for *ne*.

(ii) Sasha ne uchitel'nitsa (Sasha is not a teacher, still implying that Sasha is female).

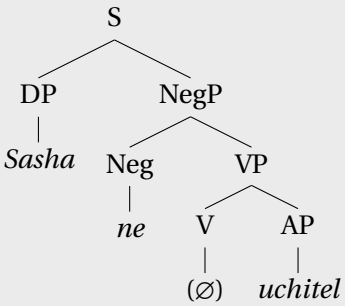


Finally, give a lexical entry for the masculine adjective so that the following sentence entails that Sasha is not a teacher and pre-

supposes that Sasha is a man. Compositionally derive the entire sentence.

(iii) Sasha ne uchitel (‘Sasha is not a teacher, implying that Sasha is a man).

- *uchitel* $\rightsquigarrow$?



Exercise 6. Gender in relational nouns.

The gender component of relational nouns like *sister* can be modelled as a presupposition using the partial operator ∂. This connects to the discussion of Strawson-symmetry in Section 2.5 of Chapter 2: treating gender as a presupposition implies that *sister* denotes a Strawson-symmetric relation, which correctly predicts that *A and B are sisters* can have a reciprocal reading.

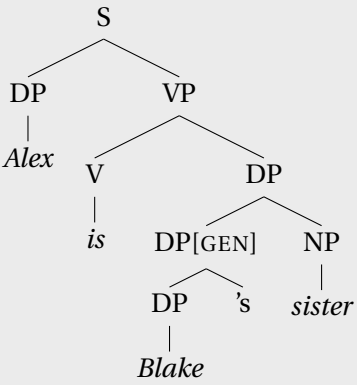
For this exercise, treat *sister* as a relational noun of type $\langle e, \langle e, t \rangle \rangle$ and use the following lexical entry for *'s* (see also the Appendix to Chapter 6):

$$\text{'s} \rightsquigarrow \lambda y \lambda R . \lambda x . R(y)(x)$$

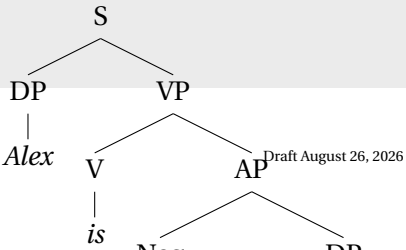
Give a lexical entry for *sister* which presupposes that one of the individuals standing in the relation is female. (Additionally, provide any other lexical entries you need to complete the trees.) Using these entries, compositionally derive the following trees.

Derive the truth conditions for both trees and verify that both presuppose that Alex is female.

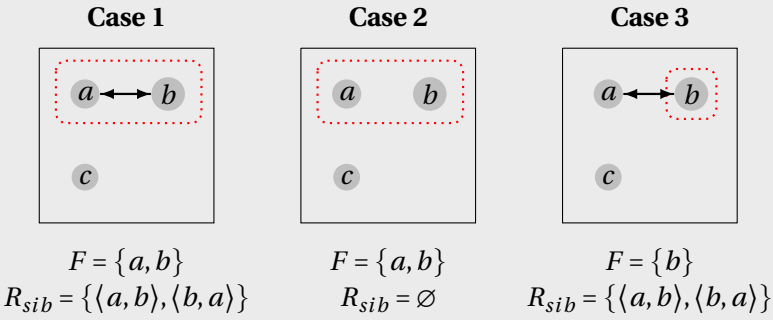
(i) Alex is Blake's sister



(ii) Alex is not Blake's sister



Using the truth conditions you derived, specify the truth value (T, F, or #) of sentences (i) and (ii) in each of the following cases. Here a = Alex, b = Blake; F is the set of female individuals (dotted red border); the sibling relation is indicated with a double arrow.



Case 1: $F = \{a, b\}$, $R_{sib} = \{\langle a, b\rangle, \langle b, a\rangle\}$

Case 2: $F = \{a, b\}$, $R_{sib} = \varnothing$

Case 3: $F = \{b\}$, $R_{sib} = \{\langle a, b\rangle, \langle b, a\rangle\}$

8.2.3 Comparison with the colon-dot notation

Readers should be aware that there is a widely used alternative notation for definedness conditions, based on Heim & Kratzer (1998). In this style of notation, the presupposition of a lambda expression is written at the beginning of the value description of that term, in between a colon and a dot. For example, (26) would be written

$$\lambda P \lambda Q : |P| = 2 . \neg \exists x [P(x) \wedge Q(x)]$$

in that notation, without any ∂ operator. For many purposes, the two styles of notation are more or less interchangeable. The colon-dot notation can in fact be adopted as syntactic sugar within our own system: '$: \phi . \psi$' abbreviates $\partial(\phi) \wedge \psi$ for type t, or more generally 'if ϕ then ψ else #' for expressions of any type.

Another advantage of the partial operator notation is that it can be used in meaning-representations that are not lambda-expressions.

For example, presuppositions can be indicated at the top of the tree for a sentence, represented by a formula.

On the other hand, the colon-dot notation is admittedly more flexible insofar as it can combine directly with expressions of types other than t; the partial operator is an expression of type t and generally combines with other expressions via conjunction of formulas. The colon-dot system allows expressions like

$$\lambda x : \mathsf{female}(x) \,.\, x$$

(the identity function restricted to females). Luckily, there is a workaround: our system provides a direct equivalent using the ι operator. In Section 8.3.2, we define notation that expresses the same idea using $\lambda x \,.\, x_{|\mathsf{female}(x)}$, or simpler yet, x^{female}.

Another point of comparison with H&K pertains not to the notation per se, but to how the notation is understood. Their treatment does not include undefined values in the ontology, instead treating a partial function as simply inapplicable to arguments outside its domain. We call this approach RADICAL UNDEFINEDNESS: rather than denoting an undefined entity, an expression whose presuppositions fail has no denotation whatsoever.

A consequence of the radical undefinedness approach is that not every syntactically well-formed expression has a denotation. LaPierre (1992) pointed out a difficulty with this type of system. He wrote, "[C]onsider a partial function g of type $\langle t, t\rangle$, ... such that $g(1)$ is undefined. Now try to apply g to $g(1)$. Strictly speaking, this application makes no sense, because $g(1)$ is not an argument at all" (p. 521). If we treated expressions containing presuppositions as radically undefined, then many of the expressions that we would be tempted to write in the course of doing compositional derivations would strictly speaking make no sense. When we use Function Application to compute the truth-and-definedness conditions of complex expressions, and put the expressions representing their meanings together using the function application syntax, putting the argument in parentheses next to the function,

we want to be making sense, even if the object language expressions whose meaning we are representing might be nonsensical. This is our reason for following LaPierre (1992) in giving the undefined a status as an object at every type, rather than adopting radical undefinedness.

8.3 Undefined values of other types

The three-valued logic developed above assigns a third truth value to propositions whose presuppositions fail. But can non-propositional expressions—expressions of type *e*, for example—also be undefined? In this section we argue that they can, examining the definite determiner and possessives as the primary cases.

8.3.1 The definite determiner

Recall that definite descriptions often convey uniqueness, as discussed earlier in Chapter 6 in relation to Generalized Quantifier theory. Suppose that we were in Sweden, and you were not entirely sure who was in the royal family, and in particular whether there were any princesses, and if there were, how many there were. Suppose then that someone were to tell you: *Guess what! I'm attending a banquet with the princess tonight.* You would probably infer that there is one and only one contextually relevant princess. (There are actually multiple princesses in Sweden, so a sincere and well-informed speaker would probably not use the expression *the princess* out of the blue. The point is that if someone were to do so, their utterance would convey that only one is relevant.) Thus definite descriptions convey EXISTENCE (that there is a relevant princess, in this case), and UNIQUENESS (that there is at most one).[5]

[5]The term *uniqueness* is used variously in the literature. Here we use it in the sense of *pure uniqueness*: the cardinality of the set of relevant individuals is at most one. Under this reading, existence and uniqueness are logically independent conditions. Some authors instead use *uniqueness* to mean *exactly one*,

The Russellian analysis of definite descriptions In "On Denoting", Russell (1905) proposes to analyze definite descriptions[6] on a par with the quantifiers we analyzed in Chapter 6. He proposes that a sentence like *The princess smokes* means 'There is exactly one princess and she smokes':

(28) $\exists x.[\text{princess}(x) \wedge \forall y.[\text{princess}(y) \rightarrow x = y] \wedge \text{smokes}(x)]$

This expression can be read as follows: There exists some x such that (i) x is a princess, and (ii) every y that is a princess is equal to x (in other words, there are no princesses other than x), and (iii) x smokes.

According to this treatment, the definite determiner introduces an entailment both that there is a princess (existence, part (i) above) and that there is only one (uniqueness, part (ii) above). The sentence is thus predicted to be false if there are either no princesses or multiple ones.

Exercise 7. Read the above formula aloud to yourself and then write out the words that you said. Which part of this formula ensures uniqueness?

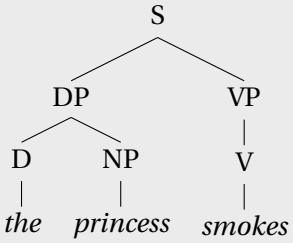
Exercise 8. (a) Give a lexical entry for *the* that yields the kind of meaning for *The princess smokes* that Russell envisions. It should combine with *princess* and *smokes* to yield (28) as a translation. Hint: Since it introduces an existential quantifier, you might look to other existential-quantifier-introducing expressions we have encountered as a model.

which entails existence; under that reading the two conditions collapse into one.

[6] Russell's primary concerns in this paper lay in the domain of philosophical logic — in particular, with the question of how we can speak meaningfully about things that do not exist, such as the present king of France, without falling into contradiction.

(b) What is the type of *the* under your treatment?

(c) Show this lexical entry in action in the following tree:



The Frege/Strawson analysis of definite descriptions Strawson (1950), in a response to Russell titled "On Referring" and building on some ideas of Frege's, agrees that definite descriptions signal existence and uniqueness of something satisfying the description, but he disagrees with Russell's proposal that these implications are entailments. His argument centers around so-called EMPTY DESCRIPTIONS: definite descriptions in which nothing satisfies the descriptive content. For example, since France is not a monarchy, *the king of France* is an empty description. Strawson writes,

> To say, "The king of France is wise" is, in some sense of "imply", to *imply* that there is a king of France. But this is a very special and odd sense of "imply". "Implies" in this sense is certainly not equivalent to "entails" (or "logically implies").

Strawson argues for this thesis as follows:

> Now suppose someone were in fact to say to you with a perfectly serious air: *The king of France is wise.* Would you say, *That's untrue?* I think it is quite certain that you would not. But suppose that he went on to ask you whether you thought that what he had just said

> was true, or was false; whether you agreed or disagreed with what he had just said. I think you would be inclined, with some hesitation, to say that you did not do either; that the question of whether his statement was true or false simply *did not arise*, because there was no such person as the king of France. You might, if he were obviously serious (had a dazed, astray-in-the-centuries look), say something like: *I'm afraid you must be under a misapprehension. France is not a monarchy. There is no king of France.*

Strawson's observation is that we feel squeamish when asked to judge whether a sentence of the form *The F is G* is true or false when there is no F. The reason is that the sentence presupposes something false: there is an attempt to refer to something that does not exist. This sort of analysis is called "Fregean", as it captures an intuition that was expressed earlier by Frege (1892). As a simple illustration, consider *the moon*: under a Fregean analysis, *the* takes as input the predicate *moon* and returns the unique individual satisfying it, if there is exactly one. If not—no moons, or multiple moons—the phrase is undefined. (We set aside plural definite descriptions like *the stars* until Chapter 9.)

Frege worked through a more complex example. One passage in which his view on the definite article comes forth involves a discussion of the expression *the negative square root of 4*. According to Frege, such an expression, like a proper name, denotes an individual (corresponding to type e in modern parlance):

> We have here a case in which out of a concept-expression, a compound proper name is formed, with the help of the definite article in the singular, which is at any rate permissible when one and only one object falls under the concept.

We assume that by "concept-expression", Frege means an expression of type $\langle e, t \rangle$, and that by "compound proper name", Frege

means a complex expression of type e. Frege's key idea is that such a use of the definite article is "permissible" only if one and only one object falls under the description. We formalize this by treating *the* as denoting a function of type $\langle\langle e,t\rangle,e\rangle$ that returns the unique satisfier of its input predicate if there is exactly one, and a special 'undefined' value otherwise.

The iota operator To implement Frege's theory, we introduce a special 'undefined individual' of type e. We use the symbol $\#_e$ to denote this individual in our meta-language. One advantage of doing so is that every expression has some semantic value or other, so our system can compute a denotation even in case of a presupposition failure. This symbol is not meant to be introduced as an expression of our logical representation language L_λ (although we could easily add a corresponding symbol to the representation language); rather we use $\#_e$ in our meta-language to refer to this 'undefined entity' we are imagining, specifying this as the denotation for empty descriptions.[7] A definite description of the form *the F* will denote $\#_e$ whenever the number of satisfiers of *F* is not exactly one.

To give a lexical entry for *the* on which it denotes $\#_e$ unless the predicate it combines with holds of exactly one individual, we introduce a new symbol into our logic:

$$\iota$$

which is the Greek letter 'iota'. Like the λ symbol, ι can bind a variable. Here is an example:

$$\iota x\,.\,P(x)$$

This is an expression of type e. It denotes the unique individual satisfying P if there is exactly one such individual, otherwise it denotes $\#_e$.

[7] Other notations that have been used for the undefined individual include Kaplan's (1977) †, standing for a 'completely alien entity' not in the set of individuals, Landman's (2004) **0**, and Oliver & Smiley's (2013) *O*, pronounced 'zilch'.

To add the ι symbol to our logic, first we add a syntax rule producing iota-expressions:

Syntax rule: Iota (to be generalized)
If ϕ is an expression of type t, and u is a variable of type e, then $\iota u\,.\,\phi$ is an expression of type e.

The semantics of iota-expressions is defined as follows:

Semantic rule: Iota (to be generalized)
If ϕ is an expression of type t, and u is a variable of type e,

$$[\![\iota u\,.\,\phi]\!]^{M,g} = \begin{cases} d & \text{if } [\![\phi]\!]^{M,g[u\mapsto d]} = \mathsf{T} \text{ but} \\ & \text{for all } d' \in D_e \text{ distinct from } d, [\![\phi]\!]^{M,g[u\mapsto d']} = \mathsf{F} \\ \#_e & \text{otherwise} \end{cases}$$

Here, d and d' are meta-variables that range over ordinary individuals (where "ordinary" excludes the undefined individual); these are the individuals in D_e. The semantic rule tells us that the ι operator picks out the unique value for the variable u in that set that verifies the condition ϕ. If there is no such value, the denotation is $\#_e$. In this way, ι encodes the existence and uniqueness conditions; $\#_e$ emerges as the denotation when one of these conditions is not satisfied.

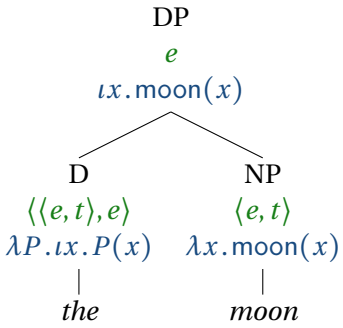
Here, we are working within an idealized semantic picture in which contextual relevance plays no further role once the model has been fixed. For the purposes of ι, the only thing that matters is the number of ordinary individuals satisfying the scope formula ϕ.

Implementing a Fregean analysis of definite descriptions With this formal tool in hand, we can now give a Fregean analysis of the definite determiner as follows:

(29) $the \rightsquigarrow \lambda P.\iota x.P(x)$

Applied to a predicate-denoting expression like $\lambda x.\mathsf{moon}(x)$, it denotes the unique moon, if there is one and only one moon in the domain of the model.

(30)

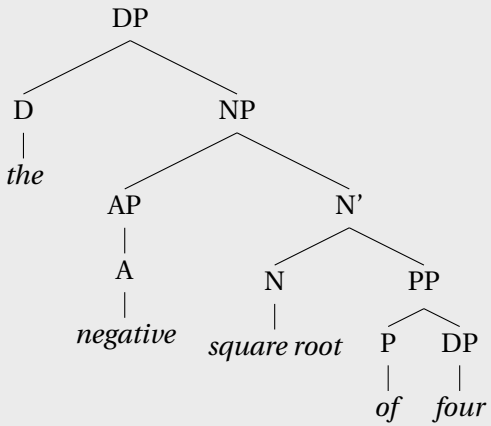


(31) $[\![\iota x.\mathsf{moon}(x)]\!]^{M,g}$

$$= \begin{cases} d & \text{if } [\![\mathsf{moon}(x)]\!]^{M,g[x\mapsto d]} = \mathsf{T} \text{ but} \\ & \text{for all } d' \in D_e \text{ distinct from } d, \\ & [\![\mathsf{moon}(x)]\!]^{M,g[x\mapsto d']} = \mathsf{F} \\ \\ \#_e & \text{otherwise} \end{cases}$$

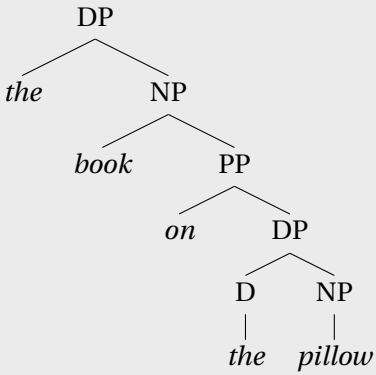
It may be useful to compare the existentially quantified formula $\exists x.\mathsf{moon}(x)$ and the term $\iota x.\mathsf{moon}(x)$. The former is a formula (type t) and the latter is a term (type e). The existentially quantified formula states that there is at least one individual that satisfies $\mathsf{moon}(x)$. If there is, its semantic value is T, otherwise F. The iota term refers successfully only if there is exactly one individual that satisfies $\mathsf{moon}(x)$. If there is, it denotes that individual, otherwise $\#_e$.

Exercise 9. Compute a derivation for the following tree according to Frege's intuitions, translating *square root* as a constant of type $\langle e,\langle e,t\rangle\rangle$, and *four* as a constant of type e:

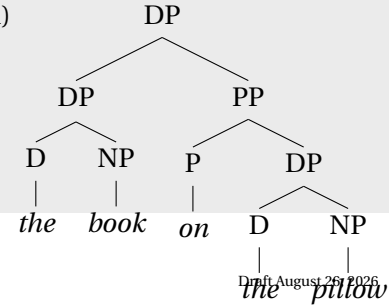


Exercise 10. Consider the two following possible parse trees for *the book on the pillow*.

(i)



(ii)



Assume the following lexical entries:

1. *book* $\rightsquigarrow \lambda x.\mathsf{book}(x)$
2. *on* $\rightsquigarrow \lambda y \lambda x.\mathsf{on}(x, y)$
3. *pillow* $\rightsquigarrow \lambda x.\mathsf{pillow}(x)$

Given these lexical entries, which of the trees above, (i) or (ii), gives the right kind of interpretation for *the book on the pillow*?

Explain your answer. You will find it helpful to annotate the nodes with their types (if not their fully beta-reduced translations as well), and consider the type at the top of the tree.

8.3.2 Gender and animacy on pronouns

Using the ι operator, we introduce the BINARY PRESUPPOSITION OPERATOR: $x_{|\phi}$, read "x, presupposing ϕ", abbreviates $\iota y.[y = x \wedge \phi]$—the unique individual identical to x that satisfies ϕ—and denotes x subject to the presupposition ϕ. This gives our system a direct counterpart to the colon-dot notation for expressions of non-t types (cf. Section 8.2.3). For example, the colon-dot expression $\lambda x : \mathsf{female}(x).x$ (the identity function restricted to females) corresponds in our system to:

$$\lambda x.x_{|\mathsf{female}(x)}$$

The $x_{|\phi}$ notation is especially convenient when the presupposition takes the form of a named predicate applied to the variable itself. In that case we can abbreviate further:

Abbreviation Convention 1. For any variable x and predicate constant f of the appropriate type:

$$x^{\mathsf{f}} \quad := \quad x_{|\mathsf{f}(x)}$$

That is, x^{f} denotes the variable x presupposed to satisfy property f. The convention generalizes across semantic types: f can be a predicate of any type of semantic object, not just individuals. We apply it here to the case of gender: $x_{|\mathsf{male}(x)} = x^{\mathsf{male}}$ is a variable presupposed to have a male referent, and $x_{|\mathsf{female}(x)} = x^{\mathsf{female}}$ presupposes a female referent.

One theory of pronouns is that they combine with a masculine, feminine, or inanimate feature:

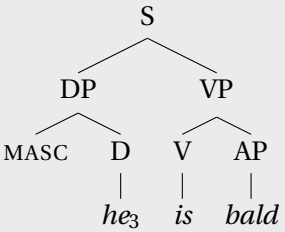
- The MASC feature combines with a pronoun to produce a type e expression, carrying the presupposition that the referent is male.
- The FEM feature combines with a pronoun to produce a type e expression, carrying the presupposition that the referent is female.
- The INAN feature combines with a pronoun to produce a type e expression, carrying the presupposition that the referent is not animate (and thus neither male nor female). We assume $\mathsf{anim}(x)$ holds iff x is animate.

Exercise 11. In the sentences below, give analyses of the FEM and INAN features following the pattern of the given MASC entry, so that the sentences carry appropriate presuppositions regarding each referent's gender or animacy. Then compositionally derive the entire trees. Note that in the negated sentences, the presupposition survives negation. Assume the following lexical entries:

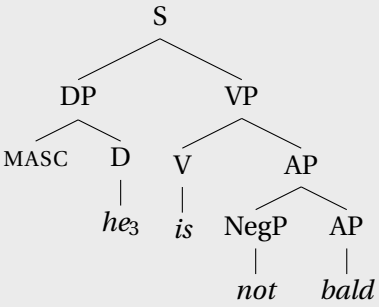
- $he_3 \rightsquigarrow v_3$, where v_3 is a variable of type e
- $she_3 \rightsquigarrow v_3$

- $it_3 \rightsquigarrow v_3$
- *bald* $\rightsquigarrow \lambda x.\mathsf{bald}(x)$, a predicate of type $\langle e, t\rangle$
- MASC $\rightsquigarrow \lambda x.x^{\mathsf{male}}$

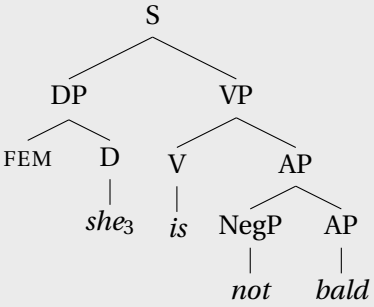
(i) He is bald



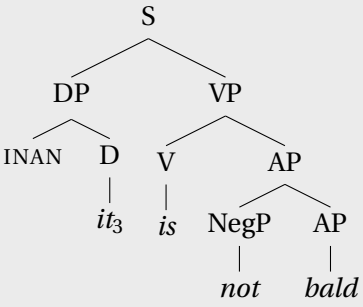
(ii) He is not bald



(iii) She is not bald



(iv) It is not bald



8.3.3 Possessives

The iota operator can be used in the analysis of other phenomena as well. We give one example here: possessives. As mentioned above, possessives trigger presuppositions:

(32) Blake loves Alex's sister.
≫ Alex has a sister.

The fact that this inference is a presupposition can be seen via the projection test; for example, *Blake doesn't love Alex's sister* also implies that Alex has a sister.

Following the appendix to Chapter 6, we treat *sister* as a relational noun of type $\langle e, \langle e, t \rangle \rangle$, and the possessive marker *'s* as:

(33) *'s* $\rightsquigarrow \lambda y \lambda R . \lambda x . R(y)(x)$

This analysis works well for PREDICATIVE uses of possessives, as in:

(34) Alex is Blake's sister.

Here *Blake's sister* denotes the predicate $\lambda x . \mathsf{sister}(x, \mathsf{b})$ (type $\langle e, t \rangle$), which applies to Alex to yield $\mathsf{sister}(\mathsf{a}, \mathsf{b})$. No uniqueness presupposition arises, since no ι operator is involved.

The analysis does not immediately extend, however, to cases where the possessive is in argument position, as in (32) or the following:

(35) Blake's sister is kind.

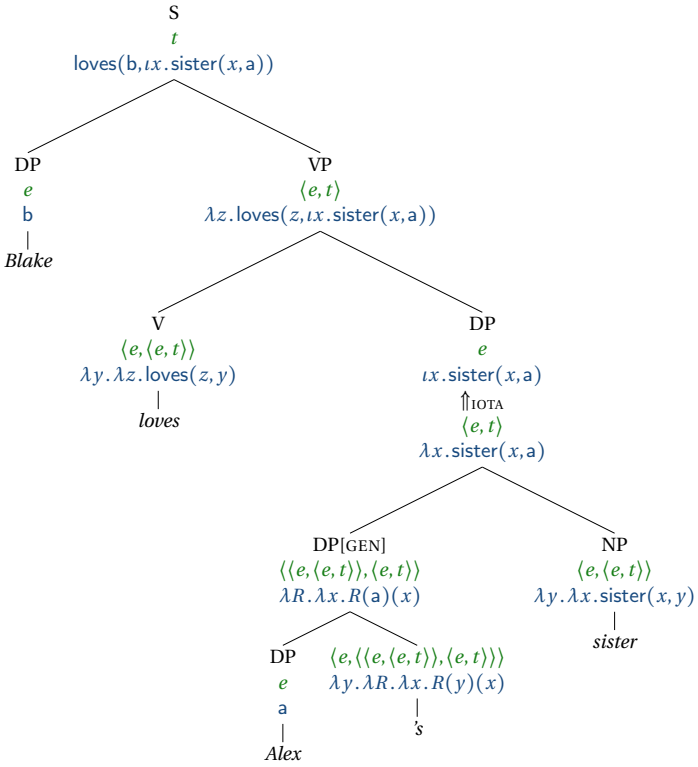
In these cases, *Alex's sister* or *Blake's sister* must denote an individual of type e, not a predicate. We bridge this gap with a silent IOTA TYPE-SHIFT, written $\Uparrow_{\text{IOTA}}$:

Type-Shifting Rule. Iota type-shift (IOTA)
If α has a translation α' of type $\langle e, t\rangle$, then α also has a translation of type e:

$$\iota x.\, \alpha'(x)$$

This operator takes a predicate of type $\langle e, t\rangle$ and returns the unique individual satisfying it, presupposing via ι that there is exactly one. The uniqueness presupposition of (32)—that Alex has exactly one sister—thus arises not from the lexical entry of *'s* but from the iota type-shift applied to *Alex's sister*. A derivation for (32) is given below:

(36)

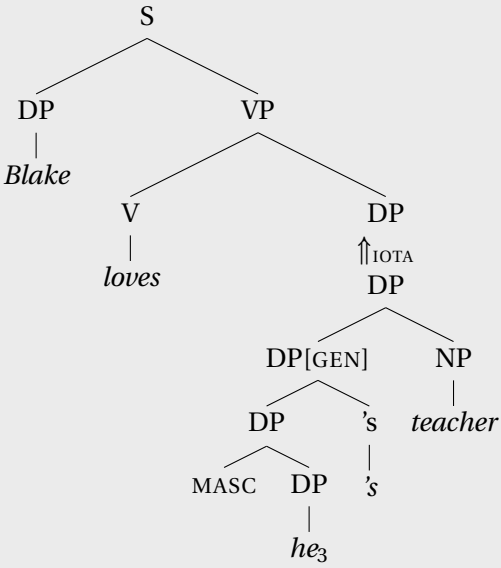


Exercise 12. Draw a derivation tree for (35), annotating each node with its translation and semantic type. Use the lexical entry in (33) and the IOTA type-shift, and treat *is kind* as a single VP of type $\langle e, t \rangle$.

Exercise 13. Possessives and free pronouns.

Compositionally derive meanings for both of the following trees.

(i) Blake loves his teacher



(ii) Blake loves her teacher
Draw the tree.

Exercise 14. Possessives and bound pronouns.

Give a binary-branching Logical Form [LF] representation for *everyone loves her teacher*, and assign truth conditions to it capturing a reading where *everyone* binds *her*. Treat *her* as a variable with a gender presupposition as in the previous section.

How many individuals in the domain are presupposed to be female according to the truth conditions that you derive? None, one, or as many as there are individuals in the domain? Additionally, how many individuals are presupposed to have exactly one teacher? Hint: look up Irene Heim's work on presuppositions of quantified sentences.

(i) Everyone loves her teacher

8.3.4 Undefined entities in the domains of functions

Definite descriptions that fail to refer, and possessives whose host NP lacks a unique possessee, both denote $\#_e$ rather than an ordinary individual. Allowing undefinedness at type e raises the question of what happens to the rest of the logic when such expressions appear as arguments to predicates, as terms in identity statements, and as the bound variables of quantifiers.

A definite description that successfully refers to something will behave just like a proper name when embedded in a larger sentence. Consider for example:

(37) The moon is spherical.

The definite description *the moon*, an expression of type e, picks out the unique moon in the given model, if there is one. Relative to a model that adheres to our earth-centric worldview and contains no moons other than that object we call *the moon*, i.e., Earth's only natural satellite, the definite description will successfully refer to it. Assuming that *spherical* is translated as an expression of type $\langle e, t\rangle$, the sentence is true in a given model M if and only if, according to M, the referent of *the moon* satisfies the predicate denoted by *spherical*. Simple enough.

But what happens if the definite description fails to refer, as in *The king of France is wise*? Let us assume that the function denoted by wise is a function from individuals to truth values, and that it maps the undefined individual $\#_e$ to the undefined truth value $\#$.[8] Since (the translation of) *the king of France*, in a model

[8] In general, we might assume that function-denoting non-logical constants have denotations that are STRICT in the sense given by LaPierre (1992): They

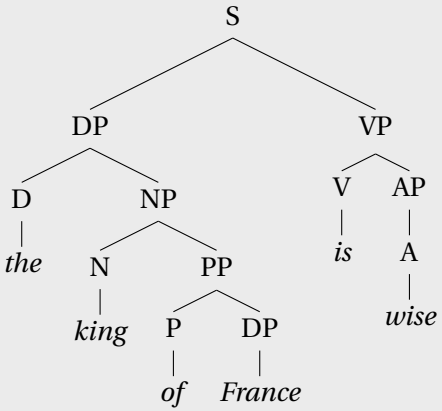
that represents the state of the world today, would have $\#_e$ as its denotation, (the translation of) *The king of France is wise* would then have $\#_t$ as its denotation. For concreteness, if the domain contains just three individuals a, b, c, the denotation of *wise* in a model M might look like:

(38) $$[\![\text{wise}]\!]^M = \begin{bmatrix} a & \to & \mathsf{T} \\ b & \to & \mathsf{F} \\ c & \to & \mathsf{T} \\ \#_e & \to & \# \end{bmatrix}$$

Exercise 15. (i) Using

- a Fregean treatment of the definite article *the*,
- a $\langle e, \langle e, t \rangle \rangle$ denotation for *king*, and
- a type e denotation for *France*

compositionally derive a translation into our representation language for the following tree:



(ii) Using the model M from (38), in which $[\![\text{wise}]\!]^M$ is as defined

map an undefined value to an undefined value.

there and *the king of France* picks out $\#_e$, what is the truth value of *The king of France is wise* in M?

The strictness assumption on predicates generalizes naturally. If a predicate of type $\langle e, t\rangle$ maps $\#_e$ to $\#$, the same logic applies to functions at every type: a function should map an undefined input to an undefined output of the appropriate type. This means we need undefined entities not just at types e and t, but at every type, including functional types. The next section formalizes this ontology and derives the semantic rules for L_∂.

8.3.5 L_∂: A partialized lambda calculus

We are now ready to give the full semantics for our new representation language L_∂. We leave the syntax of the language implicit, and just give the semantics here.

As in L_{Pred} and L_λ, the semantic values of expressions in L_∂ depend on a model and an assignment function. As usual, a model M determines a set of individuals, which we will call D, and an interpretation function I that maps each non-logical constant of type τ to a member of the fixed-up domain D^+_τ.

Again, let $\mathcal{T}$ be the set of types (e for individuals, t for truth values, $\langle e, t\rangle$ for functions from individuals to truth values, etc.). For each type $\tau \in \mathcal{T}$, the model determines a corresponding domain D_τ. Let us define the STANDARD FRAME based on D as an indexed family of sets $(D_\tau)_{\tau\in\mathcal{T}}$, where:

- $D_e = D$
- $D_t = \{\mathsf{T}, \mathsf{F}\}$
- for any types σ and τ, $D_{\langle\sigma,\tau\rangle}$ is the set of functions from D_σ to D_τ.

Following Haug (2013), we refer to these sets in the standard frame as the CLASSICAL DOMAINS. D_e is a set of individuals that does not

include the undefined entity. $D_t = \{\mathsf{T}, \mathsf{F}\}$. For functional types $\langle\sigma,\tau\rangle$, there is a classical domain $D_{\langle\sigma,\tau\rangle}$ consisting of the total functions from D_σ to D_τ.

For L_∂, along with the standard frame, models are associated with an AUGMENTED FRAME as well. Following Haug (2013), we distinguish between CLASSICAL DOMAINS D_τ and FIXED-UP DOMAINS D^+_τ, for any type τ. The fixed-up domains include the undefined entities. (In making this assumption, we are following the precedent set by LaPierre (1992), who distinguishes between the set of 'ordinary meanings' and the set of 'partial meanings'; the undefined entities are part of the latter but not the former set.)

For atomic types, the fixed-up domains are defined as the union of the classical domains with the undefined entity of the corresponding type:

- $D^+_t = (D_t \cup \{\#_t\})$.
- $D^+_e = (D_e \cup \{\#_e\})$.

For complex types $\langle\sigma,\tau\rangle$, we define $D^+_{\langle\sigma,\tau\rangle}$ as the set of functions from D^+_σ to D^+_τ. We define the undefined entity of any complex type $\langle\sigma,\tau\rangle$ as a function whose output is always $\#_\tau$, the undefined entity of type τ. In other words, $\#_{\langle\sigma,\tau\rangle}(k) = \#_\tau$ for any element k of D^+_σ. Since $\#_{\langle\sigma,\tau\rangle}$ is a function from D^+_σ to D^+_τ, it is a member of $D^+_{\langle\sigma,\tau\rangle}$. An AUGMENTED FRAME based on D as an indexed family of sets $(D_\tau)^+_{\tau\in\mathcal{T}}$ characterized in this way.

The truth of a universally quantified formula should not require that the undefined entity make the scope formula true. For instance, if $D_e = \{a, b\}$ and a and b both have property happy in model M, then $\forall x.\mathsf{happy}(x)$ should count as true in M, even though $\mathsf{happy}(x)$ might not take on the value T when an assignment function maps x to $\#_e$. *Don't let the undefined entity ruin the party!* So in the semantic rule for universal quantification, the undefined entity should be excluded from consideration as a relevant witness.

For type e, exclusion of the undefined entity could be achieved by appealing to the classical domain instead of the fixed-up domain. But for functional types like $\langle e, t\rangle$, this solution will not work.[9] With a type like $\langle e, t\rangle$, the classical domain $D_{\langle e,t\rangle}$ is not a subset of the fixed-up domain $D^{+}_{\langle e,t\rangle}$. Suppose D_e is just the set $\{a, b\}$. While $D_{\langle e,t\rangle}$ is composed entirely of functions whose domain is just that set, $D^{+}_{\langle e,t\rangle}$ is composed entirely of functions whose domain is the set $\{a, b, \#_e\}$. This means that there is nothing in $D_{\langle e,t\rangle}$ that is also in $D^{+}_{\langle e,t\rangle}$! In fact, the two sets are disjoint. The variables in our language will in general range over objects in the augmented domain, and thus for functional types will never take on values in the classical domain. We therefore cannot define the semantics for universal quantification merely by restricting the set of relevant witnesses to the classical domain, as long as we want universal quantification to range over variables of functional types.

There are multiple options we could consider in order to deal with this issue. One would be to simply restrict universal quantification to type e. But we would like to have quantification over other types as well, as motivated in Chapter 5 (see (45a)–(45b)). The solution we adopt here is simply to exclude just the undefined entity of the relevant type. We introduce the following convention:

(39) **Convention:** D^{-}_{τ} stands for $D^{+}_{\tau} - \{\#_\tau\}$.

D^{-}_{τ} can be referred to as the RESTRICTED DOMAIN for type τ. One possible worry about this solution is that it might not be restricted enough; for functional types, it leaves *in* all manner of functions involving undefined entities that have no correlate in the classical domain. We leave it as an exercise to set up a satisfactory system

[9] The system presented in Haug (2013) falls prey to this issue. A system presented in a previous version of this book does as well. We are grateful to Ede Zimmermann, David Beaver, and Edgar Onea for extremely valuable discussion of this issue.

that excludes them (see Exercise 19). We are not currently aware of any data that bears on whether or not they should be excluded.

We now give the semantic rules for quantified sentences. Consider the following sentence:

(40) Every boy loves his sister.

Ignoring the presupposition of *every*, and building on the analysis of possessive *'s* given above (i.e. decomposing *his* into *he* + *'s* and treating *his sister* as synonymous with *the sister of him*), this would be translated:

(41) $\forall x.[\mathsf{boy}(x) \rightarrow \mathsf{loves}(x, \iota y[\mathsf{sister}(y,x)])]$

This formula will be true in a model where every element of D_e satisfies the following formula, when plugged in for x:

(42) $\mathsf{boy}(x) \rightarrow \mathsf{loves}(x, \iota y[\mathsf{sister}(y,x)])$

What, precisely, is the presupposition of (40)? The iota expression will have a defined value just in case there is exactly one entity y satisfying the following description:

(43) $\mathsf{sister}(y,x)$

How many entities are there satisfying this description? Well, it depends on what individual x is assigned to. Suppose there is a boy with no sisters. Call him Freddie. If x is assigned to Freddie, then the iota expression will denote the undefined individual. Does that suffice to make the universal claim as a whole undefined? Or can we still say that the universal claim is true as long as every boy *who has exactly one sister* loves that sister? In other words, if the assortment of truth values that the open formula *x loves his sister* (the SCOPE FORMULA) takes on as we cycle through various values for x contains both T and # (but never F), should the truth value for the proposition *Every boy loves his sister* (the UNIVERSAL PROPOSITION) be T or should it be #? Different authors

have advocated different answers to this question.

Given that a universal claim can be seen as a big conjunction, and an existential claim can be seen as a big disjunction, it is natural to set up a partial logic in such a way that universal quantifiers 'match' conjunction, and existential quantifiers 'match' disjunction. With a Weak Kleene treatment of conjunction, then, this desideratum leads to the following treatment of universal quantification:

Semantic rule: Universal quantification
If u is a variable of type τ, and ϕ a formula:

$$[\![\forall u.\phi]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \mathsf{T} \text{ for all } k \in D_\tau^- \\ \# & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \# \text{ for some } k \in D_\tau^- \\ \mathsf{F} & \text{otherwise} \end{cases}$$

With this treatment of the universal quantifier, a universal claim is false only if (a) the scope proposition never takes on an undefined value for any ordinary value of the variable, and (b) the scope proposition is false for at least one ordinary value of the variable (where "ordinary" excludes the undefined entity). For example, *Every boy loves his sister* is false only if every boy has exactly one sister and at least one boy doesn't love his sister.

Exercise 16. Consider a model with a boy named Freddie who has no sisters, and two other boys. Both of the other boys have exactly one sister that they love. Under the treatment of universal quantification just given, what truth value does (41) have in this model?

In classical predicate logic, the universal and existential quantifiers are duals of each other; in particular, $\forall x.\neg\phi$ is equivalent to $\neg\exists x.\phi$. To maintain this equivalence, given the treatment of

the universal quantifier just given, we must define the existential quantifier as follows:

Semantic rule: Existential quantification
If u is a variable of type τ, and ϕ a formula:

$$[\![\exists u\,.\,\phi]\!]^{M,g} = \begin{cases} \mathsf{F} & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \mathsf{F} \text{ for all } k \in D_\tau^- \\ \# & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \# \text{ for some } k \in D_\tau^- \\ \mathsf{T} & \text{otherwise} \end{cases}$$

So an existential claim is true only if the scope proposition is true for at least one member of the domain and is never undefined. For example, *Some boy loves his sister* is false if every boy has exactly one sister (and no boy loves his sister).

Exercise 17. A famous example in the literature on presupposition is Heim's (1983) *A fat man was riding his bicycle*. Intuitively, this sentence does not presuppose that every fat man owns a bicycle; it suffices for a single fat man to own one. How does it turn out in the theory we have developed so far? What does our theory predict this sentence to presuppose? To answer this question, start by giving a compositional derivation for the sentence. You may treat *fat man* and *was riding* as single words. Then discuss in what models the truth conditions you derive would be defined.

The exercise above illustrates a general limitation of Weak Kleene: the existential quantification rule makes an existential undefined if *any* witness makes the scope undefined—even when another witness makes it true. So *A fat man was riding his bicycle* comes out undefined whenever any fat man (not just "the one being described") lacks a bicycle, predicting a spuriously strong presupposition.

The assertion operator **A** (introduced in Section 8.4) provides a remedy. Inserting **A** within the quantifier's scope turns unde-

fined instances into F instead, so that a fat man without a bicycle makes *his* instance of the scope false rather than undefined. Under Beaver & Krahmer's (2001) Floating-A Theory, this interpretation is available when pragmatic principles favor it, correctly predicting that the sentence need not presuppose that every fat man owns a bicycle.

A related issue concerns identity. Under what circumstances do we want to say that a given sentence of the form $\alpha = \beta$ is true, given that α or β might denote $\#_e$? We certainly don't want it to turn out to be the case that *The king of France is the Grand Sultan of Germany* is a true statement. In keeping with the void/nonsense interpretation of the third truth value, and assuming that any equality statement involving an undefined value is nonsense, we assume the following, for expressions α and β of any type τ:

Semantic rule: Identity

- If $[\![\alpha]\!]^{M,g}$ and/or $[\![\beta]\!]^{M,g}$ is $\#_\tau$, then $[\![\alpha = \beta]\!]^{M,g} = \#$.
- If neither is undefined, then:
  $[\![\alpha = \beta]\!]^{M,g} = \begin{cases} \mathsf{T} \text{ if } [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g} \\ \mathsf{F} \text{ otherwise.} \end{cases}$

Under this treatment, *The king of France is the Grand Sultan of Germany* comes out as undefined, rather than true.

So much for models and the various sorts of domains: classical, augumented, and restricted. An assignment function for L_∂ is a total function whose domain consists of the variables of the language such that if u is a variable of type τ then $g(u) \in D^+_\tau$.

The semantic rules for L_∂ are the following. Relative to a model M and an assignment function g, the semantic values of expressions in L_∂ are defined as follows.

1. **Basic Expressions**

(a) If α is a non-logical constant, then $[\![\alpha]\!]^{M,g} = I(\alpha)$.

(b) If α is a variable, then $[\![\alpha]\!]^{M,g} = g(\alpha)$.

2. **Iota**
If u is a variable of type τ, and ϕ a formula:

$$[\![\iota u.\phi]\!]^{M,g} = \begin{cases} d & \text{if } [\![\phi]\!]^{M,g[u\mapsto d]} = \mathsf{T} \text{ but} \\ & \text{for all } d' \in D_\tau^- \text{ distinct from } d, [\![\phi]\!]^{M,g[u\mapsto d']} = \mathsf{F} \\ \#_\tau & \text{otherwise} \end{cases}$$

3. **Application**
If α is an expression of type $\langle\sigma,\tau\rangle$, and β is an expression of type σ, then $[\![[\alpha(\beta)]]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$

4. **Identity**
If α and β are expressions of type τ, then $[\![\alpha = \beta]\!]^{M,g}$

$$= \begin{cases} \# & \text{if } [\![\alpha]\!]^{M,g} = \#_\tau \text{ or } [\![\beta]\!]^{M,g} = \#_\tau \\ \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

5. **Connectives**

(a) The connectives $\wedge$, $\vee$, and $\neg$, have a Weak Kleene semantics, as defined as in Table 8.2.

(b) The binary connectives $\rightarrow$ and $\leftrightarrow$ are defined in terms of $\neg$ and $\vee$:

- $[\phi \rightarrow \psi]$ is defined as $[\neg\phi \vee \psi]$
- $[\phi \leftrightarrow \psi]$ is defined as $[[\phi \rightarrow \psi] \wedge [\psi \rightarrow \phi]]$

Hence the truth tables for $\rightarrow$ and $\leftrightarrow$ are as in Table 8.3.

(c) Semantics for the unary connective ∂ are defined as in Table 8.4. A second unary connective **A**, the assertion operator, is introduced in Section 8.4.

6. **Quantification**

   (a) If u is a variable of type τ, and ϕ a formula:

$$[\![\forall u.\phi]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \mathsf{T} \text{ for all } k \in D_\tau^- \\ \# & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \# \text{ for some } k \in D_\tau^- \\ \mathsf{F} & \text{otherwise} \end{cases}$$

   (b) If u is a variable of type τ, and ϕ a formula:

$$[\![\exists u.\phi]\!]^{M,g} = \begin{cases} \mathsf{F} & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \mathsf{F} \text{ for all } k \in D_\tau^- \\ \# & \text{if } [\![\phi]\!]^{M,g[u\to k]} = \# \text{ for some } k \in D_\tau^- \\ \mathsf{T} & \text{otherwise} \end{cases}$$

7. **Lambda Abstraction**
   If α is an expression of type τ and u a variable of type σ then $[\![\lambda u.\alpha]\!]^{M,g}$ is that function h from D_σ^+ into D_τ^+ such that for all objects k in D_σ^+, $h(k) = [\![\alpha]\!]^{M,g[u\mapsto k]}$.

This is just one example of a complete system; other design choices are also possible.

One happy consequence of the way things have been set up is worth pointing out. It follows from the definitions just given that an expression of type $\langle\sigma,\tau\rangle$ denotes a function from D_σ^+ to D_τ^+. This means in particular that $\#_\sigma$ is in the domain of any function denoted by an expression of type $\langle\sigma,\tau\rangle$. Making this assumption helps to ensure that eta-reduction is valid. For example, let P be a constant of type $\langle t,t\rangle$ and let p be a variable of type t. Suppose P denoted one of the four unary classical truth functions, and did not have the undefined entity in its domain. On the other hand, if we assume that lambda expressions denote functions that can accept undefined entities as inputs, then $\lambda p.P(p)$ denotes a function that is also defined on the nonclassical truth value $\#$. So P would not be equivalent to $\lambda p.P(p)$, in violation of the eta-reduction principle. Letting P accept undefined values as input allows us to maintain the eta-reduction principle.

It is similarly important that assignment functions are able to map variables to the undefined entity; this ensures that beta-reduction is valid. To see why, consider the assertion operator **A** (introduced in Section 8.4), which maps # to F. Assume that $[\![\phi]\!]^{M,g}$ is #. Then $[\![\mathbf{A}(\phi)]\!]^{M,g}$ is F. Let p be a variable of type t. If the only possible values that p could take on via an assignment function were T and F, then $[\lambda p.\mathbf{A}(p)](\phi)$ would be undefined even though $\mathbf{A}(\phi)$ is false. Hence beta-reduction would not be valid; it wouldn't always be the case that $[\lambda u.\alpha](\beta)$ is equivalent to a version of α with all free instances of u substituted for β.

Exercise 18. In M_1, there are no elevators. In M_2, there is one, namely e_1. In M_3, there are two, namely e_2 and e_3. Specify the indicated semantic values:

(a) $[\![\iota x.\mathsf{elevator}(x)]\!]^{M_1}$

(b) $[\![\exists x.\mathsf{elevator}(x)]\!]^{M_1}$

(c) $[\![\partial(\exists x.\mathsf{elevator}(x))]\!]^{M_1}$

(d) $[\![\iota x.\mathsf{elevator}(x)]\!]^{M_2}$

(e) $[\![\exists x.\mathsf{elevator}(x)]\!]^{M_2}$

(f) $[\![\partial(\exists x.\mathsf{elevator}(x))]\!]^{M_2}$

(g) $[\![\iota x.\mathsf{elevator}(x)]\!]^{M_3}$

$\rightarrow$	T	F	#
T	T	F	#
F	T	T	#
#	#	#	#

$\leftrightarrow$	T	F	#
T	T	F	#
F	F	T	#
#	#	#	#

Table 8.3: Truth tables for $\rightarrow$ and $\leftrightarrow$ in Weak Kleene logic

(h) $[\![\exists x.\mathsf{elevator}(x)]\!]^{M_3}$

(i) $[\![\partial(\exists x.\mathsf{elevator}(x))]\!]^{M_3}$

Exercise 19. Our definition of D_τ^- in the semantics for quantification fails to exclude many functions that have no correlate in the classical domain. One might think that this is not a desirable feature of the system, and really, at some conceptual level, universal quantification should only be sensitive to members of the classical domain. Indeed, Ede Zimmermann has suggested to us (p.c.) that this might be the right way to go. Redefine D_τ^- to exclude these spurious cases. The result of your definition should be such that D_τ^- is always a subset of the D_τ^+, and the cardinality of D_τ^- is equal to D_τ. You can do this by defining a unique mapping from D_τ to D_τ^+, assigning a representative in the augmented domain for each member of the classical domain.

Then revise the definitions for universal and existential quantification so that, for each type, only the elements of the ordinary domain are relevant witnesses.

8.4 Accommodation

Not all presuppositions react the same way in discourse when they go unsatisfied. Imagine you are talking with someone you've just met and they say:

(44) My boyfriend always says that to me.

You have no prior information about whether this person has a boyfriend, yet you seamlessly update your understanding of the conversation to include the existence of one. You don't pause to

object; you simply go along with it. This is GLOBAL ACCOMMODATION: the presupposed content is silently added to the common ground, as if it had been asserted just before the utterance (Lewis, 1979).

How easily accommodation proceeds depends on plausibility. Suppose you work in an office and have no idea whether your boss Malcolm owns any animals. Your co-worker tells you:

(45) Malcolm loves his elephant.

You would likely be surprised and react with something like *Hey, wait a minute! I didn't know Malcolm owns an elephant.* The presupposed content is too conspicuous to slip into the common ground unnoticed. But suppose instead your co-worker says:

(46) Malcolm loves his cat.

You would probably take this in stride, adding it to your beliefs without raising a fuss—treating (46) as if it meant:

(47) Malcolm owns a cat, and he loves it.

The implications of (45) and (46)—that Malcolm has an elephant and a cat, respectively—are presuppositions of these sentences, as the projection test confirms. If a sentence with an unsatisfied presupposition is neither true nor false, then (45) and (46) should behave alike in discourse. But their effects are quite different. PRESUPPOSITION ACCOMMODATION is a process by which hearers update the common ground with the presupposed content of a sentence, so that the sentence can be understood in a context where its presuppositions hold. Through accommodation, an unsatisfied presupposition—which would ordinarily make a sentence neither true nor false—can instead make it simply false, as if the presupposed content had been asserted beforehand.

Once accommodation is sanctioned as a theoretical possibility, a much wider range of interpretations is available for any sen-

tence containing a presupposition trigger. In general, more careful argumentation is required to diagnose presuppositions and distinguish them empirically from entailments.

8.4.1 Local accommodation and the assertion operator

The accommodation cases above are GLOBAL: the presupposed content is added to the common ground and the sentence is then evaluated against the updated context. But there is a subtler kind, in which accommodation happens locally, within the semantic composition of the sentence itself.

Consider the definite description *the king of France*, which presupposes a unique king exists (Section 8.3). In a neutral context, a hearer might globally accommodate (48):

(48) The king of France is bald.

But now consider:

(49) The king of France isn't bald — there is no king of France!

Here the speaker is not just denying the baldness; they are canceling the presupposition itself. The sentence sounds true and informative: the second clause explains the first. Under our standard treatment, however, (49) comes out #. The negation of (48), analyzed as $\neg[\partial(\exists!x.\mathsf{king}(x)) \wedge \mathsf{bald}(\iota x.\mathsf{king}(x))]$, is # by Weak Kleene whenever the presupposition fails. Something is wrong.

The problem is that the speaker intends a reading in which the presupposition is not projected but absorbed into the at-issue content that is being negated. This is LOCAL ACCOMMODATION: the presupposed content is treated as ordinary asserted content within a sub-part of the sentence, so the sentence has a classical truth value even when the presupposition would otherwise fail.

Beaver & Krahmer (2001) propose a formal device that models this effect: the ASSERTION OPERATOR $\mathbf{A}$ (Table 8.4). Like ∂, $\mathbf{A}$ is a unary operator of type $\langle t, t\rangle$. Unlike ∂, it maps every non-

	∂
T	T
F	#
#	#

	A
T	T
F	F
#	F

Table 8.4: Truth tables for the partial operator and the assertion operator

true value—including #—to F rather than preserving it. In effect, $\mathbf{A}(\phi)$ asserts ϕ without presupposing anything.

Letting ϕ abbreviate the translation of (48), the two readings of (49) are:

(50) a. $\neg\phi$ (standard external negation: # when no king)
b. $\neg\mathbf{A}(\phi)$ (local accommodation: $\neg$F = T when no king)

Reading (50b)b, in which **A** is inserted before negation applies, correctly delivers T in a world without a king of France: it treats the presupposition as at-issue content and asserts that the whole thing is false. This is the local accommodation reading.

It is sometimes useful to think of the semantic contribution of a sentence as a conjunction of two components: the presupposition π and the AT-ISSUE CONTENT ϕ (the "main point"). When implemented using ∂, one conjoins $\partial(\pi)$ with ϕ. It is crucial here to use a Weak Kleene conjunction: $[\partial(\pi) \wedge \phi]$ is # whenever π fails, regardless of ϕ. The assertion operator promotes the presupposition to an ordinary entailment: $\mathbf{A}(\partial(\pi) \wedge \phi)$ is T if both π and ϕ are T, and F otherwise.

8.5 The projection problem

The treatment of presupposition we have given so far correctly predicts that presuppositions can PROJECT: If a sentence S is embedded in a larger sentence S′, and S carries a presupposition, then S′ may carry the same presupposition. For instance, both

of the following sentences convey that there are multiple candidates:

(51) a. Every candidate is qualified.
b. It is not the case that every candidate is qualified.

We have set up our logic so that $\neg\phi$ has the truth value $\#$ whenever ϕ has that truth value. So, whenever (51a) is undefined, (51b) is undefined as well. Hence, if a given sentence S presupposes some sentence P—in the sense that the truth value of S is undefined unless P is true—then the negation of S is predicted to presuppose P as well. In that sense, the presupposition is predicted to PROJECT OVER NEGATION, under the theoretical assumptions we have laid out. In fact, given our use of the Weak Kleene connectives, presuppositions are *always* predicted to project from an embedded sentence to a more complex one containing it (unless we include Beaver & Krahmer's (2001) assertion operator **A**).

But presuppositions do not always project. Consider the following examples:

(52) If there is a king of France, then the king of France is wise.

(53) Either there is no king of France or the king of France is wise.

Neither of these sentences as a whole implies that there is a king of France. The problem of determining when a presupposition projects is called the PROJECTION PROBLEM.

The expressions *if*/*then* and *either*/*or* are FILTERS, which do not let all presuppositions "through", so to speak. (Imagine the presuppositions floating up from deep inside the sentence, and getting trapped when they meet *if*/*then* or *either*/*or*.) Being filters, operators like *if*/*then* and *either*/*or* do let some presuppositions through. Examples:

(54) If France remains neutral, then the king of France is wise.

(55) Either France is lucky or the king of France is wise.

Both of these sentences carry the presupposition that there is a king of France. The key difference between (54) and (52) is that in (54), the antecedent (*France remains neutral*) does not entail the consequent's presupposition (that there is a king of France). Similarly, in (55), unlike in (53), the first disjunct (*France is lucky*) does not entail the second disjunct's presupposition (again, that there is a king of France).

In general, when the antecedent of the conditional (the *if*-part) entails a presupposition of the consequent (the *then*-part), the presupposition gets filtered out, so the larger, complex sentence does not carry the presupposition. With a disjunction, the generalization is that presupposition of one disjunct gets filtered out when the negation of another disjunct entails it.

We have used the word *entail* in the generalizations above. In (52) and (53), the part of the sentence that is supposed to entail the presupposition is simply equivalent to the presupposition. But it could also be stronger, and entail the presupposition without being equivalent to it:

(56) a. If France is a constitutional monarchy with a king and a queen, then the king of France is wise.
b. Either France has not recently crowned a king for the first time in centuries, or the king of France is wise.

In (56a), the antecedent (*France is a constitutional monarchy with a king and a queen*) is stronger than the consequent's presupposition, yet the presupposition is still filtered out. Likewise in (56b). The antecedent need not be identical to the presupposition; an entailment relation suffices. Projection failures of this kind can be formally captured within the static framework developed here using Beaver & Krahmer's (2001) assertion operator **A**; we refer the reader to their paper for details.

8.6 Toy fragment

The following is an updated version of the toy fragment introduced in Chapter 6, extended with the additions from Chapters 7 and 8.

Representation language
Chapters 6 and 7 used the language L_λ. Chapter 8 extends it to L_∂, a three-valued (partial) lambda calculus, by adding two new expression-forming devices:

Syntax rule: Definedness conditions
If ϕ is an expression of type t, then $\partial(\phi)$ is an expression of type t.

Semantic rule: Definedness conditions
$$[\![\partial(\phi)]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\phi]\!]^{M,g} = \mathsf{T} \\ \# & \text{otherwise.} \end{cases}$$

Syntax rule: Iota
If ϕ is an expression of type t and u is a variable of type e, then $\iota u.\phi$ is an expression of type e.

Semantic rule: Iota
$$[\![\iota u.\phi]\!]^{M,g} = \begin{cases} d & \text{if } [\![\phi]\!]^{M,g[u\mapsto d]} = \mathsf{T} \text{ but for all } d' \in D_e \\ & \text{distinct from } d,\ [\![\phi]\!]^{M,g[u\mapsto d']} = \mathsf{F} \\ \#_e & \text{otherwise} \end{cases}$$

Type conventions

- Variables of type e: x, y, z, and indexed variants x_i
- Variables of type $\langle e, t\rangle$: P, Q
- Variables of type $\langle\langle e, t\rangle, t\rangle$: Z
- Variables of type $\langle e, \langle e, t\rangle\rangle$: R
- Variables of type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: f
- Variables of type t: p, q
- Constants of type e: a, b, c
- Constants of type $\langle e, t\rangle$: smiled, laughed, smokes, drinks, kind, swedish, happy, proud, sailor, pirate, singer, drummer, musician, person, male
- Constants of type $\langle e, \langle e, t\rangle\rangle$: loves, likes, hugged, with, sister, teacher

Variables of types not listed above carry an explicit type subscript (e.g., x_a for a variable of schema type a). When multiple variable symbols appear in the same formula, they are understood to be distinct from one another and from any indexed variant (x_i) in the same formula.

Syntax

S	→	DP VP
S	→	SNeg S
VP	→	V (DP|AP|PP)
AP	→	A (PP)
DP	→	D (NP)
DP	→	DP[GEN] NP
DP[GEN]	→	DP Poss
NP	→	N (PP)
NP	→	A NP

NP	→	NP CP
CP	→	DP_i [WH] S
PP	→	P DP

Schemas (where *XP* ranges over any phrasal category):

XP	→	Neg *XP*

Syntactic transformations Both syntactic transformations from the fragment in Chapter 7 (relative clause formation, QR) carry over.

Composition rules All composition rules from the fragment in Chapter 7 carry over.

Type-shifting operations

- **RAISE-O** (Object Raising)
  If $\alpha \rightsquigarrow \alpha'$ of type $\langle e, \langle a, t\rangle\rangle$, then α also has a translation of type $\langle\langle\langle e, t\rangle, t\rangle, \langle a, t\rangle\rangle$:

  $$\lambda Z \lambda x_a . Z(\lambda y . \alpha'(y)(x))$$

- **RAISE-S** (Subject Raising)
  If $\alpha \rightsquigarrow \alpha'$ of type $\langle a, \langle e, t\rangle\rangle$, then α also has a translation of type $\langle a, \langle\langle\langle e, t\rangle, t\rangle, t\rangle\rangle$:

  $$\lambda y_a \lambda Z . Z(\lambda x_e . \alpha'(y)(x))$$

- **FREE-R**
  Type-shifts a sortal noun from type $\langle e, t\rangle$ to type $\langle e, \langle e, t\rangle\rangle$ by introducing a free relation variable R:

  $$\text{FREE-R} \rightsquigarrow \lambda P . \lambda y . \lambda x . [P(x) \wedge R(y)(x)]$$

  Written $\Uparrow_{\text{FREE-R}}$ on the shifted node.

- **REL**
  Lifts a predicate modifier of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ to operate on relational nouns, producing type $\langle\langle e,\langle e,t\rangle\rangle,\langle e,\langle e,t\rangle\rangle\rangle$:

  $$\text{REL} \rightsquigarrow \lambda f.\lambda R.\lambda y.f(R(y))$$

  Written $\Uparrow_{\text{REL}}$ on the shifted node.

- **IOTA** (Iota type-shift)
  Applied to a possessive NP in argument position, of type $\langle e,t\rangle$, produces an expression of type e encoding a uniqueness presupposition:

  $$\Uparrow_{\text{IOTA}} \rightsquigarrow \lambda P.\iota x.P(x)$$

Lexical entries

V
smiled, laughed, smokes, drinks follow $W \rightsquigarrow \lambda x.W(x)$; *hugged* follows $V \rightsquigarrow \lambda y\lambda x.V(x,y)$.

- *is* $\rightsquigarrow \lambda P.P$
- *loves* $\rightsquigarrow \lambda y\lambda x.\mathsf{loves}(x,y)$
- *likes* $\rightsquigarrow \lambda y\lambda x.\mathsf{likes}(x,y)$

A
Swedish, happy, kind, proud follow $W \rightsquigarrow \lambda x.W(x)$.

N
sailor, pirate, singer, drummer, musician follow $W \rightsquigarrow \lambda x.W(x)$.

- *sister* $\rightsquigarrow \lambda y.\lambda x.\mathsf{sister}(x,y)$
- *teacher* $\rightsquigarrow \lambda y.\lambda x.\mathsf{teacher}(x,y)$

C

- *that* $\rightsquigarrow \lambda p.\, p$

D (proper names)

- *Alex* $\rightsquigarrow \mathsf{a}$
- *Blake* $\rightsquigarrow \mathsf{b}$
- *Casey* $\rightsquigarrow \mathsf{c}$

D (determiners)

- *a* $\rightsquigarrow \lambda P.\, P$
- *some* $\rightsquigarrow \lambda P \lambda Q.\, \exists x.\, [P(x) \wedge Q(x)]$
- *no* $\rightsquigarrow \lambda P \lambda Q.\, \neg \exists x.\, [P(x) \wedge Q(x)]$
- *every* $\rightsquigarrow \lambda P \lambda Q.\, [\partial(\exists x.\, P(x)) \wedge \forall x.\, [P(x) \rightarrow Q(x)]]$
- *neither* $\rightsquigarrow \lambda P \lambda Q.\, [\partial(|P| = 2) \wedge \neg \exists x.\, [P(x) \wedge Q(x)]]$
- *the* $\rightsquigarrow \lambda P.\, \iota x.\, P(x)$
- *one* $\rightsquigarrow \lambda P.\, \lambda Q.\, [\exists! x.\, [P(x) \wedge Q(x)]]$

D (quantificational noun phrases)

- *somebody* $\rightsquigarrow \lambda P.\, \exists x.\, [\mathsf{person}(x) \wedge P(x)]$
- *nobody* $\rightsquigarrow \lambda P.\, \neg \exists x.\, [\mathsf{person}(x) \wedge P(x)]$
- *everybody* $\rightsquigarrow \lambda P.\, [\partial(\exists x.\, \mathsf{person}(x)) \wedge \forall x.\, [\mathsf{person}(x) \rightarrow P(x)]]$
- *something* $\rightsquigarrow \lambda P.\, \exists x.\, P(x)$
- *nothing* $\rightsquigarrow \lambda P.\, \neg \exists x.\, P(x)$
- *everything* $\rightsquigarrow \lambda P.\, \forall x.\, P(x)$

Poss

- *'s* $\rightsquigarrow \lambda y \lambda R . \lambda x . R(y)(x)$

P

- *of* $\rightsquigarrow \lambda x . x$
- *with* $\rightsquigarrow \lambda y \lambda x . \mathsf{with}(x, y)$

Neg

- *not* $\rightsquigarrow \lambda P \lambda x . \neg P(x)$

SNeg

- *It is not true that* $\rightsquigarrow \lambda p . \neg p$

Features

- MASC $\rightsquigarrow \lambda x . x^{\mathsf{male}}$

Syncategorematically interpreted items

The following items have no independent lexical denotation; their interpretation is determined entirely by the composition rules above.

D WH (relative pronouns): *who, which* — interpreted via Predicate Abstraction

Dem SIM: *such* — interpreted via Predicate Abstraction

D (pronouns): *he, she, him, her, it, they, them* — interpreted as x_i via the Pronouns and Traces Rule

9 Coordination and plurals

9.1 Coordination

Let us now consider coordination in more detail. We may include sentences with *and* and *or* among the well-formed expressions of our language by extending our syntax and lexicon as follows:

(1) **Syntax**

S	→	S &S
&S	→	& S

(2) **Lexicon**

&: *and*, *or*

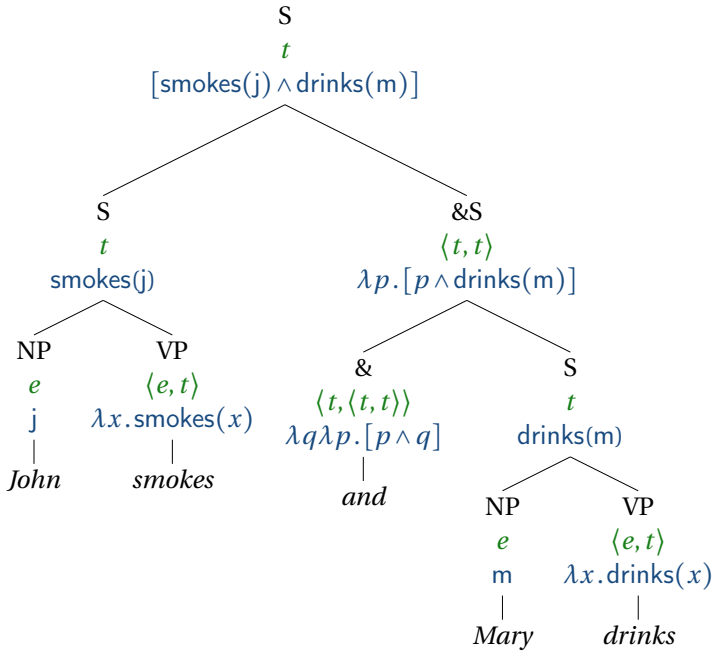
To translate these into the lambda calculus, we can simply write the following (here, p and q are variables over truth values):

(3) a. $and_S \rightsquigarrow \lambda q \lambda p . [p \wedge q]$

b. $or_S \rightsquigarrow \lambda q \lambda p . [p \vee q]$

This will work for coordinations of sentences. For example, here is a tree for *John smokes and Mary drinks*:

(4)



Sentences are not the only kinds of expressions that can be coordinated, though. Here are a few examples:

(5) a. Somebody smokes and drinks. (VP and VP)
b. No man and no woman laughed. (DP and DP)
c. Susan caught and ate the fish. (V and V)

It is clear that we need to extend our grammar. Since these examples do not cover all the possibilities, it will not do to introduce fixes to the syntax and semantics one at a time. Instead, we need to formulate a general pattern and then extend our syntax and semantics according to it.

How shall we analyze the semantics of coordination? An early style of analysis consisted in analyzing all coordinations as underlyingly sentential, even those of constituents other than sentences. For example, VP coordination was analyzed as involving

deletion of the subject of the second sentence (indicated here as strikethrough):

(6) a. John smokes and drinks.
b. John smokes and ~~John~~ drinks.

It was soon found that this would not work. If VP coordination really was sentential coordination in disguise, then all VP coordinations should be semantically equivalent to their sentential relatives. This may be the case for simple sentences, as above. But quantifiers break this equivalence. The following two sentences are *not* paraphrases, as their translations into logic show.

(7) a. Somebody smokes and drinks.
$\exists x.[\mathsf{smokes}(x) \wedge \mathsf{drinks}(x)]$
b. Somebody smokes and somebody drinks.
$[\exists x.\mathsf{smokes}(x) \wedge \exists x.\mathsf{drinks}(x)]$

(8) a. Everybody smokes or drinks.
$\forall x.[\mathsf{smokes}(x) \vee \mathsf{drinks}(x)]$
b. Everybody smokes or everybody drinks.
$[\forall x.\mathsf{smokes}(x) \vee \forall x.\mathsf{drinks}(x)]$

Exercise 1. For each of the two sentence pairs above, establish that they are not equivalent by describing a scenario in which one of them is true and the other one is false.

Luckily, it is also possible to design a grammar in which coordinated constituents are directly generated syntactically, and directly interpreted semantically. We can extend the syntax with a single pair of schema rules, where &*XP* denotes the phrase headed by the coordinator:

(9) **Syntax**

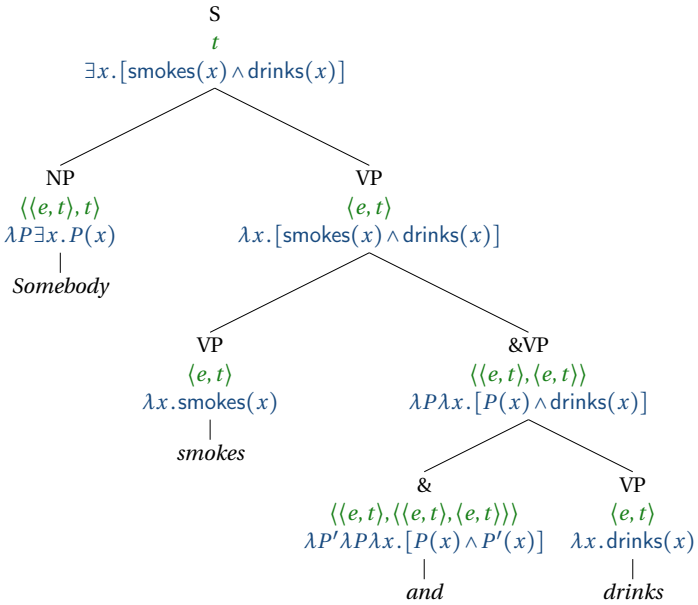
XP	→	*XP* &*XP*
&*XP*	→	& *XP*

The semantic side is trickier. It is not obvious if we can give a single denotation for each conjunction that covers all of its uses across categories. So we will first look at a few cases individually, and then generalize over them. For VP coordination, the following entries for *and* and *or* will do:

(10) a. $and_{VP} \rightsquigarrow \lambda P' \lambda P \lambda x . [P(x) \wedge P'(x)]$
b. $or_{VP} \rightsquigarrow \lambda P' \lambda P \lambda x . [P(x) \vee P'(x)]$

This tree shows the entry for *and* in action. The result is what we want: the quantifier *somebody* takes scope over *and*.

(11)



What about coordinations of transitive verbs, as in *Sue loves and hates John*? Assuming that transitive verbs translate to expressions of type $\langle e, \langle e, t \rangle \rangle$, that is, (Schönfinkelized) binary relations, the version of *and* that should be used to coordinate them should take two binary relations and return a new binary relation. The

following entries will do that trick.

(12) a. $and_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . [R(y)(x) \wedge R'(y)(x)]$
b. $or_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . [R(y)(x) \vee R'(y)(x)]$

Given *loves* and *hates*, these lexical entries will produce a new relation, 'loves and hates'.

Exercise 2. Using the lexical entry for *and* above, draw the tree for *Susan caught and ate the fish*.

Noun phrase coordination (that is, coordination of DPs) can be approached in the same way. Let us first look at coordinations of quantifiers:

(13) a. Every man and every woman laughed.
$\forall x . [\mathsf{man}(x) \rightarrow \mathsf{laughed}(x)] \wedge$
$\forall x . [\mathsf{woman}(x) \rightarrow \mathsf{laughed}(x)]$
b. A man or a woman laughed.
$\exists x . [\mathsf{man}(x) \wedge \mathsf{laughed}(x)] \vee$
$\exists x . [\mathsf{woman}(x) \wedge \mathsf{laughed}(x)]$

Since quantifiers have a higher type, they take verb phrases as arguments. This makes the entries for *and* and *or* very similar to their VP-coordinating counterparts:

(14) a. $and_{DP} \rightsquigarrow \lambda Q' \lambda Q \lambda P . [Q(P) \wedge Q'(P)]$
b. $or_{DP} \rightsquigarrow \lambda Q' \lambda Q \lambda P . [Q(P) \vee Q'(P)]$

Exercise 3. Using the lexical entries above, draw the trees for *Every man and every woman laughed* and *A man or a woman laughed*.

In all of the examples so far, the two constituents being coordinated were of the same semantic type. That is not always the

case. As the following example shows, a type-e noun phrase like *John* can be coordinated with a type-$\langle\langle e,t\rangle,t\rangle$ noun phrase.

(15) John and every woman laughed.

The translation we should obtain for this sentence is as follows:

$$[\mathsf{laughed}(\mathsf{j}) \wedge [\forall x.\mathsf{woman}(x) \rightarrow \mathsf{laughed}(x)]]$$

In order to be able to reuse the lexical entry above, and in order to avoid deviating from the pattern we have established so far, we will adjust the type of *John* to make it equal to that of *every woman*. For this purpose, we introduce a new type-shifting rule that introduces a possible translation of type $\langle\langle e,t\rangle,t\rangle$ for every translation of type e:

Type-Shifting Rule. Entity-to-quantifier shift
If $\alpha \rightsquigarrow \alpha'$, where α' is of type e, then α can also be translated as follows:

$$\lambda P.P(\alpha')$$

This rule, which goes back to Montague (1974b), is also called Montague-lift. It encapsulates the insight that an individual x can be recast as the set of all the properties that x has. Essentially, the rule inverts the predicate-argument relationship between the subject and the verb phrase of a sentence. For example, if *John* $\rightsquigarrow j$ then also *John* $\rightsquigarrow \lambda P.P(j)$. That translation is of type $\langle\langle e,t\rangle,t\rangle$. It denotes a function that maps predicates to truth values. Any predicate that holds of John is within the characteristic set of this function. In a sentence like *John laughed*, this function takes the verb phrase denotation as an argument. In a sentence like *John and every woman laughed*, we are able to conjoin this function with *every woman* using the entry and_{DP}. The resulting coordinated DP denotation can combine with any verb phrase, or if it

occurs in nonsubject position, the resulting type mismatch can be repaired using the mechanisms from Chapter 7 (either QR or further type-shifting).

Exercise 4. Draw the tree for *John and every woman laughed* and derive a semantic interpretation for it compositionally.

Exercise 5. Draw a tree for *John and Sue smoke* and give a derivation that results in:

$$[\text{smokes}(\text{j}) \wedge \text{smokes}(\text{s})]$$

You will need to apply the type shifter once on each conjunct.

We are now ready to generalize over syntactic categories. This is done by defining a single operator $\sqcap$ that generalizes over all these categories and then translating *and* as $\sqcap$ (and similarly for disjunction and a corresponding operator $\sqcup$). All of the entries for conjunction and for disjunction have $\wedge$ and $\vee$ at their core respectively. And all of them operate on types that end in t, namely $\langle e, t\rangle$ for VP coordination, $\langle e, \langle e, t\rangle\rangle$ for coordination of transitive verbs, and $\langle\langle e, t\rangle, t\rangle$ for DP coordination. The following recursive definitions will work for every type that ends in t.

(16) $$\sqcap_{\langle\tau,\langle\tau,\tau\rangle\rangle} = \begin{cases} \lambda q \lambda p\,.\, p \wedge q & \text{if } \tau = t \\ \lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1}\,.\, \sqcap_{\langle\sigma_2,\langle\sigma_2,\sigma_2\rangle\rangle}(X(Z))(Y(Z)) & \text{if } \tau = \langle\sigma_1,\sigma_2\rangle \end{cases}$$

(17) $$\sqcup_{\langle\tau,\langle\tau,\tau\rangle\rangle} = \begin{cases} \lambda q \lambda p\,.\, p \vee q & \text{if } \tau = t \\ \lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1}\,.\, \sqcup_{\langle\sigma_2,\langle\sigma_2,\sigma_2\rangle\rangle}(X(Z))(Y(Z)) & \text{if } \tau = \langle\sigma_1,\sigma_2\rangle \end{cases}$$

For more details on this approach, see for example Partee & Rooth (1983) and Winter (2001).

Here is how the schema in (16) derives DP-coordinating *and*. The type of DP (after lifting entities to quantifiers if necessary) is $\langle\langle e,t\rangle,t\rangle$. So the type of DP-coordinating *and* is $\langle\tau,\langle\tau,\tau\rangle\rangle$, where $\tau = \langle\langle e,t\rangle,t\rangle$. Since $\tau \neq t$, we look for σ_1 and σ_2 such that $\tau = \langle\sigma_1,\sigma_2\rangle$. This works for $\sigma_1 = \langle e,t\rangle$ and $\sigma_2 = t$. We plug in these definitions into the last line of (16) and get:

(18) $\lambda X_{\langle\langle e,t\rangle,t\rangle}\lambda Y_{\langle\langle e,t\rangle,t\rangle}\lambda Z_{\langle e,t\rangle}.\sqcap_{\langle t,\langle t,t\rangle\rangle}(X(Z))(Y(Z))$

To resolve $\sqcap_{\langle t,\langle t,t\rangle\rangle}$, we apply Definition (16) once more. This time, $\tau = t$, so the result is simply logical conjunction:

(19) $\sqcap_{\langle t,\langle t,t\rangle\rangle} = \lambda q\lambda p.p \wedge q$

We plug this into the previous line and get the final result:

(20) $\lambda X_{\langle\langle e,t\rangle,t\rangle}\lambda Y_{\langle\langle e,t\rangle,t\rangle}\lambda Z_{\langle e,t\rangle}.Y(Z) \wedge X(Z)$

This is indeed equivalent to our entry for DP-coordinating *and* in (14a). The entry will only work if both DPs are of type $\langle\langle e,t\rangle,t\rangle$. If necessary, one or both DPs may need to be lifted into that type first by applying the type shifter above.

Exercise 6. Show how the schema can be applied to VP-coordination.

9.2 Collective predication and mereology

All of the occurrences of *and* that we have seen so far can be related to the denotation of logical conjunction. This is not always the case, though. Consider the following example.

(21) John and Mary are a happy couple.

There is no obvious way to formulate the truth conditions of (21) using logical conjunction. It cannot be represented as:

$$[\mathsf{happy\text{-}couple(j)} \wedge \mathsf{happy\text{-}couple(m)}]$$

since this would entail $\mathsf{happy\text{-}couple(j)}$ as well as $\mathsf{happy\text{-}couple(m)}$. In other words, it would have the entailments that John is a happy couple and that Mary is a happy couple. These are obviously nonsensical because a singular individual can't be a couple. Only two people can form a happy couple. Predicates like *be a couple* are called COLLECTIVE. They apply to collections of individuals directly, without applying to those individuals. In this sentence, then, the word *and* does not seem to amount to logical conjunction but to the formation of a collection, in this case, the "collective individual" John-and-Mary.

Another example of collective predication was given by Link (1983), at the beginning of his paper. He writes:

> The weekly *Magazine* of the German newspaper *Frankfurter Allgemeine Zeitung* regularly issues Marcel Proust's famous questionnaire which is answered each time by a different personality of West German public life. One of those recently questioned was Rudolf Augstein, editor of *Der Spiegel*; his reply to the question: ["Which property of your friends do you appreciate the most?"] was ... "that they are few".
>
> Clearly, this is not a property of any one of Augstein's friends; yet, even apart from the *esprit* it was designed to display the answer has a straightforward interpretation. The phrase ... predicates something *collectively* of a *group* of objects, here: Augstein's friends.

To talk about such collections, we need to extend our formal setup. On the semantic side, we will add collections of individuals to our model. You might suspect that we would represent

these collections as sets, so that *John and Sue* would be represented as the set that contains just these two individuals. Instead, we will extend our formal toolbox by borrowing from MEREOLOGY, the study of parthood. There are many reasons for this choice. One is that using mereology for this purpose has been standard practice in formal semantics since Link (1983). Another reason is that set theory makes formal distinctions that turn out not to be needed in mereology. Where set theory is founded on two relations ($\in$ and $\subseteq$), mereology collapses them into one, the parthood relation. This relation holds both between John and John-and-Sue (where in set theory, we would use $\in$), and also between John-and-Sue and John-and-Sue-and-Mary (where in set theory, we would use $\subseteq$). Mereology also provides an operator, $\oplus$, that allows us to put individuals together to form collections. The formal objects that represent these collections in mereology are called SUMS. For example, the collection John-and-Sue is represented formally as John$\oplus$Sue. This sum is not a set, but rather an individual, albeit an individual that has parts. Collective predicates apply directly to such sums. For example, in the sentence *John and Sue are a happy couple*, the phrase *happy couple* can be translated as a predicate of type $\langle e, t \rangle$ that can be truthfully predicated of the sum John$\oplus$Sue, as that sum is an object in the domain corresponding to type e.

In mereology, the domain can be organized into an algebraic structure. An algebraic structure is essentially a set with a binary operation (in this case, the part-of relation) defined on it. Figure 9.1 illustrates such a structure. The circles stand for the individual Andrews Sisters Maxine, LaVerne, and Patty, and for the sums that are built up from them. We will use the word INDIVIDUAL to range over all the circles in this structure. We will refer to Maxine, LaVerne, and Patty, as ATOMIC INDIVIDUALS; the other circles stand for individuals which are not atomic. In mereology, the terms ATOM and ATOMIC refer to anything which does not have any parts other than itself; they are technical terms that do not necessarily coincide with physical or metaphysical notions of what is an atom. For

semantic purposes, it is common to assume that individuals that can be described with a singular count noun are atoms. By this criterion, any human being is treated as an atom; so is any hand, and any committee, with no mereological parthood relation holding between these entities (as opposed to the matter that constitutes them).

(22) **Definition: Atom**
$\mathsf{atom}(x) \stackrel{\text{def}}{=} \neg\exists y[y \sqsubset x]$
(An atom is an individual with no proper parts.)

In Figure 9.1, the lines between the circles stand for the parthood relations that hold between the various individuals. We will assume that parthood is reflexive, transitive, and antisymmetric, or as it is called in mathematics, a "partial order". Reflexivity means that everything is part of itself. Reflexivity is an axiom that governs the notion of parthood:

(23) **Axiom of reflexivity**
$\forall x[x \sqsubseteq x]$
(Everything is part of itself.)

Reflexivity may not be intuitive but it is a mere formal convenience, and it can be eliminated by defining a distinct notion of proper parthood: *a* is a proper part of *b* just in case *a* is both part of and distinct from *b*.

(24) **Definition: Proper part**
$x \sqsubset y \stackrel{\text{def}}{=} x \sqsubseteq y \wedge x \neq y$
(A proper part of a thing is a part of it that is distinct from it.)

(So the lines in the diagram represent the *proper* parthood relations among the individuals depicted.)

Transitivity means that if *a* is part of *b* and *b* is part of *c*, then *a* is also part of *c*. Among the axioms of the system we develop in

this chapter is that the parthood relation is transitive:

(25) **Axiom of transitivity**
$\forall x \forall y \forall z [x \sqsubseteq y \wedge y \sqsubseteq z \rightarrow x \sqsubseteq z]$
(Any part of any part of a thing is itself part of that thing.)

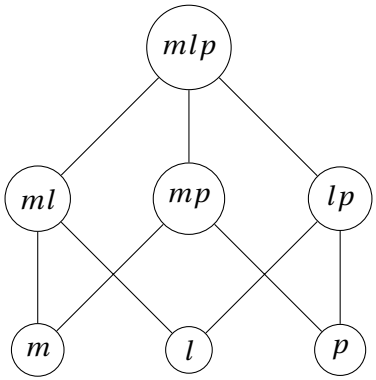
For example, according to Figure 9.1, Maxine is part of Maxine⊕LaVerne, and Maxine⊕LaVerne is part of Maxine⊕LaVerne⊕Patty; therefore, by transitivity, Maxine is also part of Maxine⊕LaVerne⊕Patty.

Finally, antisymmetry means that two distinct things cannot both be part of each other.

(26) **Axiom of antisymmetry**
$\forall x \forall y [x \sqsubseteq y \wedge y \sqsubseteq x \rightarrow x = y]$
(Two distinct things cannot both be part of each other.)

This condition is very intuitive. For example, since Maxine is part of Maxine⊕LaVerne, it follows that Maxine⊕LaVerne is not part of Maxine. Together, the axioms of reflexivity, transitivity, and antisymmetry constrain parthood to be a partial order.

Figure 9.1: An algebraic structure.
Abbreviations: m = Maxine; l = LaVerne; p = Patty.



The notion of sum can be defined in terms of this parthood relation as follows:[1]

(27) **Binary sum**
$x \oplus y$ = that individual k such that
- $x \sqsubseteq k$,
- $y \sqsubseteq k$,
- there is no $k' \sqsubset k$ such that $x \sqsubseteq k'$ and $y \sqsubseteq k'$.

So far we have been using the symbols $\oplus$ and $\sqsubseteq$ as part of the meta-language, because we are referring directly to the corresponding operation and relation that our models now provide. We will also have a way of referring to these concepts in the representation language. The new representation language will be an extension of L_∂ that supports talk of parts and sums; let us refer to it as L_*.

Models for our L_* furnish a part-of relation $\sqsubseteq$, which in turn determines a sum operation $\oplus$, along with a domain and an interpretation function. The parthood symbol and sum symbols can be imported into our representation language as follows:

Semantic rule: Parthood
If α and β are expressions of type e and M determines the individual-part relation $\sqsubseteq$:

$$[\![\alpha \sqsubseteq \beta]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g} \sqsubseteq [\![\beta]\!]^{M,g} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

[1] This definition can be stated more rigorously using the notion of OVERLAP: two individuals x and y OVERLAP, written $x \circ y$, if they share a common part: $x \circ y \stackrel{\text{def}}{=} \exists z[z \sqsubseteq x \wedge z \sqsubseteq y]$. Using overlap, the sum $x \oplus y$ can be characterized following Tarski (1929) as the individual k such that (i) $x \sqsubseteq k$ and $y \sqsubseteq k$, and (ii) for all z, if $z \sqsubseteq k$ then $z \circ x$ or $z \circ y$. Condition (ii) ensures that k contains nothing beyond what is needed to include x and y.

Semantic rule: Sum
If α and β are expressions of type e and M determines the sum operation $\oplus$:

$[\![\alpha \oplus \beta]\!]^{M,g} = [\![\alpha]\!]^{M,g} \oplus [\![\beta]\!]^{M,g}$.

Exercise 7. Formulate an additional lexical entry for and_{DP} that conjoins two entities of type e and returns their sum. Draw the tree for *John and Sue met*.

9.3 The plural

9.3.1 Algebraic closure

Coordinations of proper nouns are not the only way to talk about sums:

(28) a. Maxine, LaVerne and Patty met.
b. Some singers met.
c. Three singers met.
d. The singers met.

In each of these three sentences, the collective predicate *met* applies to a sum x. Only (28a) fully specifies the parts of that sum, while (28b) through (28d) describe it partially. That is, the sentence specifies that they are all singers, but not their precise identities.

Just like the predicate *met*, the plural noun *singers* can be seen as denoting a predicate that applies to the sum x. What is the denotation of the noun *singers*? One way to describe it is in terms of the conditions it imposes on x, namely, *singers* requires it to be the

sum of some singers. In general, the denotation of a plural noun can be described in terms of the denotation of its corresponding singular noun. If we take P to be the set of all the entities in the denotation of the singular noun, then the plural noun denotes the set that contains any sum of things taken from P.

In order to make this notion precise, we need a way of talking about the sum of a set of things. So far, all we have is a binary sum operator $\oplus$ that puts together two individuals. The sum of any nonempty set of individuals S is always the lowest individual that sits above every element of S. (This corresponds to the mathematical notion of "least upper bound".) For example, if S consists of the two atomic individuals Maxine and LaVerne, then the lowest individual that sits above these two is Maxine⊕LaVerne. Sometimes the sum of S can be a member of S. For example, if S consists of Maxine and Maxine⊕LaVerne, Maxine⊕LaVerne again. And if S consists of just one individual, such as Patty, then its sum is that individual itself (or himself or herself). The sum operation that applies to sets is sometimes called GENERALIZED SUM (as opposed to binary sum). We use $\bigoplus S$ to denote the sum of a set S, and define it as follows:

(29) **Definition: Generalized sum**
Relative to a given model M determining the part relation $\sqsubseteq$, given a nonempty set $S \subseteq D_e$,

$\bigoplus S \stackrel{\text{def}}{=}$ the individual k in D_e such that:
(i) for all $x \in S$, $x \sqsubseteq k$;
(ii) there is no $k' \sqsubset k$ such that for all $x \in S$, $x \sqsubseteq k'$.

As the use of the definite article *the* in this definition suggests, this definition depends on the fundamental assumption that any nonempty set of individuals has one and exactly one sum.

It will be useful to have a symbol of the representation language corresponding to the notion of generalized sum that applies to expressions of type $\langle e, t \rangle$ and produces new expressions

of that type.

Semantic rule: Generalized sum
If π is an expression of type $\langle e, t\rangle$, then:
$[\![\oplus \pi]\!]^{M,g} = \bigoplus\{x \mid [\![\pi]\!]^{M,g}(x) = \top\}$

Every expression of the form $\oplus\pi$ is of type e, denoting a possibly-plural individual. Such expressions therefore act as terms; they can serve as the argument to a predicate, and we will see examples of this below.

The plural morpheme can be treated semantically using the notion of algebraic closure, which builds on the notion of generalized sum. The ALGEBRAIC CLOSURE ${}^*S$ of a set S is the set that contains, for each non-empty subset S' of S, the sum of S'.

(30) **Definition: Algebraic closure**
${}^*S \stackrel{\text{def}}{=} \{x \mid \text{there exists } S' \text{ s.t. } S' \neq \varnothing \text{ and } S' \subseteq S \text{ and } x = \bigoplus S'\}$

We define a corresponding symbol in the representation language that applies to expressions of type $\langle e, t\rangle$ and produces a new expression of type $\langle e, t\rangle$ denoting the predicate that holds of an individual if it is in the algebraic closure of the set characterized by the input predicate.

Semantic rule: Star operator
If π is an expression of type $\langle e, t\rangle$, then:
$[\![{}^*\pi]\!]^{M,g}$ is that function f such that for all $x \in D_e$,
$f(x) = \top$ iff $x \in {}^*\{y \mid [\![\pi]\!]^{M,g}(y) = \top\}$

This symbol can be referred to as the STAR OPERATOR.

A simple theory of the plural morpheme *-s* as in *girls* is that it takes as input a predicate of individuals and outputs a new predicate that holds of any individual in the algebraic closure of the

input predicate. In other words, the plural introduces a star operator:

(31) **Lexical entry: Plural**
-s $\rightsquigarrow \lambda P . {}^{*}P$

For example, suppose that we are in a model with just three singers, Maxine, LaVerne and Patty. Then the denotation of the noun *singer* might be modeled as $\{m, l, p\}$, where m is short for Maxine, l is short for LaVerne, and p is short for Patty. The denotation of the noun *singers* is (loosely speaking) the algebraic closure of that set, which is the structure depicted in Figure 9.1:

$$\{m, l, p, m \oplus l, m \oplus p, l \oplus p, m \oplus l \oplus p\}$$

This set contains everything that is either a singer or a sum of two or more singers. It might seem strange to include individual singers in this set. After all, it sounds strange to say *Maxine are singers*, and the sentence *Some doctors are in the room* is false if only one doctor is in the room. And indeed, Link himself proposed excluding them. But this leads to a different problem: It makes *singers* essentially synonymous with *two or more singers*. But this leads to the wrong predictions in downward-entailing environments. For example, *No doctors are in the room* is not synonymous with *No two or more doctors are in the room*. Consider the case where a single doctor is in the room. Here only one of the two sentences is true. For this reason it is common to continue to use (31) as the denotation of the plural (Krifka, 1986), and to rule out *Maxine are singers* on pragmatic grounds. That is, *singers* literally means *one or more singers*. Sentences like **John is singers* and **John are singers* are assumed to be ruled out on syntactic or pragmatic rather than on semantic grounds.

Link gave plural individuals the status of first-class citizens in the logical representation of natural language. That is, they belong to D_e and are not treated differently from atomic individuals. This allowed him to represent collective predicates like *meet* as

predicates that apply directly to sum individuals:

(32) a. Maxine, LaVerne, and Patty met. $\rightsquigarrow \mathsf{meet}(\mathsf{mx} \oplus \mathsf{l} \oplus \mathsf{p})$
b. Some singers met. $\rightsquigarrow \exists x.[{}^{*}\mathsf{singer}(x) \wedge \mathsf{meet}(x)]$

As seen in (32a), Link represented sentential conjunction in a different way than noun phrase conjunction. This has the consequence that even the translations of equivalent sentences can look very different:

(33) a. Maxine is a singer and LaVerne is a singer.
$\rightsquigarrow [\mathsf{singer}(\mathsf{mx}) \wedge \mathsf{singer}(\mathsf{l})]$
b. Maxine and LaVerne are singers.
$\rightsquigarrow {}^{*}\mathsf{singer}(\mathsf{mx} \oplus \mathsf{l})$

Exercise 8. Give the semantic type of each of the following expressions. Here, b and c are constants of type e; x, y, z are variables of type e; and P is a variable of type $\langle e, t \rangle$.

1. $\lambda x.{}^{*}\mathsf{boy}(x)$
2. x
3. P
4. $P(x)$
5. $\lambda x.P(x)$
6. $\exists x.P(x)$
7. $P(b \oplus c)$
8. $b \oplus c$
9. $(x \oplus y) \oplus z$
10. $\bigoplus \mathsf{kid}$

11. $^*\mathsf{kid}$

12. $^*(^*P)$

Exercise 9. Draw trees for the sentences in (32) and (33b), using the appropriate entries for *and* in each case. You can use the same entry for *some* as in Chapter 6. Assume that *is*, *a* and *are* denote identity functions, or treat them as vacuous nodes. Make sure that the result is as in (32) and (33b).

9.3.2 Numerals

The plural morpheme gives count nouns a cumulative denotation, but says nothing about the number of atoms involved. Numeral modifiers like *two* or *three* add a cardinality condition. We treat numerals as predicates of type $\langle e, t\rangle$ using the constant card of type $\langle e, n\rangle$, where n is the basic type for natural numbers[2] and $\mathsf{card}(x)$ returns the count of atomic parts of x:

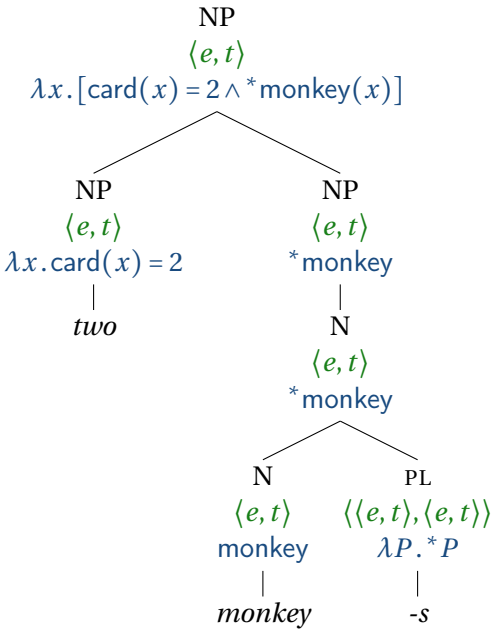
(34) **Lexical entry: Numerals**
two $\rightsquigarrow \lambda x.\mathsf{card}(x) = 2$, *three* $\rightsquigarrow \lambda x.\mathsf{card}(x) = 3$, etc.

A numeral combines with a plural noun via Predicate Modification, yielding a predicate that holds of all sums with the right cardinality and composition. The derivation for *two monkeys* is:

(35) two monkeys $\rightsquigarrow \lambda x.[\mathsf{card}(x) = 2 \wedge {}^*\mathsf{monkey}(x)]$

[2] Treating n as a basic type is a simplification: it assumes the domain D_n contains the natural numbers as objects but does not equip them with arithmetic structure — there is no ordering, no addition, and no successor function defined within L_*.

(36)



The result holds of any individual consisting of exactly two monkey-atoms.

As an NP of type $\langle e,t\rangle$, a phrase like *two monkeys* needs a determiner to become a full DP argument. When no overt determiner appears — as in bare argument uses like *Two monkeys arrived* — we posit a silent existential determiner $\varnothing_D$:

(37) $\varnothing_D \rightsquigarrow \lambda P \lambda Q . \exists x . [P(x) \wedge Q(x)]$

This gives *Two monkeys arrived* the translation

$$\exists x[\mathsf{card}(x) = 2 \wedge {}^*\mathsf{monkey}(x) \wedge \mathsf{arrived}(x)]$$

meaning there is a sum of exactly two monkeys that arrived.[3] When the overt determiner *the* is present, on the other hand, the NP

[3] Bare plurals without a numeral — *monkeys arrived, monkeys are mammals* — raise harder questions. The silent existential $\varnothing_D$ handles the indefinite reading of *monkeys arrived*, but bare plurals also have generic readings (*monkeys*

must combine with a more complex semantics — a topic we turn to next.

9.3.3 Plural definite descriptions

Supposing that *singers* denotes a predicate that holds of any singer-plurality, what does *the singers* denote? If *the singers* is translated as:

$$\iota x\,.\,{}^{*}\mathsf{singer}(x)$$

then a presupposition failure will arise as long as there is more than one singer, because more than one individual will satisfy the predicate ${}^{*}\mathsf{singer}$.[4] How can this problem be remedied?

One possible solution is to give a different kind of analysis for plural *the*, where it refers to the *sum* of the individuals that satisfy that predicate given by the noun, rather than the *unique* individual that satisfies it.

The plural definite article can then be treated as denoting this sum operator:

(38) $\text{the}_{\text{SUM}} \rightsquigarrow \lambda P\,.\,\bigoplus P$

Later we will consider a different version of *the*; the subscript SUM is there to distinguish this version of *the* from the other one we will consider. Combined with *singers*, this yields:

(39) $\text{the}_{\text{SUM}}\ \text{singers} \rightsquigarrow \bigoplus {}^{*}\mathsf{singer}$

In a model where the singers are Maxine, LaVerne and Patty, the predicate ${}^{*}\mathsf{singer}$ holds of $\{m, l, p, m\oplus l, m\oplus p, l\oplus p, m\oplus l\oplus p\}$. As

eat bananas) and kind readings (*monkeys are endangered*) that resist a simple existential analysis. Carlson (1977) treats bare plurals as kind-denoting expressions; Chierchia (1998) develops a cross-linguistic type-shifting framework. We set these complications aside here.

[4] This was pointed out by Sharvy (1980).

can be checked with Figure 9.1, the sum of this set — and therefore the denotation of (39) — is just $m \oplus l \oplus p$. This seems like a sensible denotation.

But in other cases, such as *the singer* and *the two singers*, we run into a problem. As we saw in the previous section, *two singers* translates via Predicate Modification as:

(40) two singers $\rightsquigarrow \lambda x.[\mathsf{card}(x) = 2 \wedge {}^*\mathsf{singer}(x)]$

In our model, the set characterized by *two singers* is $\{m \oplus l, m \oplus p, l \oplus p\}$.

Exercise 10. Using the silent determiner $\varnothing_D$ from (37), give a full derivation for *Two singers met.* Make sure to include $\varnothing_D$ in the tree diagram.

Now, what about *the two singers*? If we use the_{SUM} from above, then we will get the sum of the two-singer pluralities. As a glance at Figure 9.1 will confirm, this sum is $m \oplus l \oplus p$. We have applied the condition $\mathsf{card}(x) = 2$ only to the pluralities being summed up, and not to the result of this summing-up operation. So we end up with the rather odd prediction that *the two singers* refers to this sum!

Exercise 11. Translate The_{SUM} *singers met* and The_{SUM} *two singers met.*

Intuitively, *the two singers* should give rise to a presupposition failure, because there are three singers in our model. We must build a source of presupposition failure into our denotation for the plural definite. Let us therefore interpret *the P* as the single individual of which *P* holds that contains every other individual

of which P also holds:[5]

(41) $\text{the}_{\text{SUPR}} \rightsquigarrow \lambda P \iota x[P(x) \wedge \forall y[P(y) \rightarrow y \sqsubseteq x]]$

We call it this because under this theory, *the* denotes the SUPREMUM of the P's: the unique P (if there is one) that contains all other P's. In structures like the one depicted in Figure 9.1, we can check whether a given set of individuals P has a supremum by checking whether there is an element of P that sits above every element of P other than itself. This is the same procedure as the one for determining the sum of P, with one exception: in the case of the supremum of P, we check if the result is an element of P, while in the case of the sum of P, we skip this check.

It turns out that this representation even works for the singular definite article. In any model where there is exactly one singer, the set denoted by *singer* is a singleton, and since everything is part of itself, the representation in (41) picks out the only member of that singleton. In all other models, the ι operator will not be defined.

Exercise 12. Translate *The*$_{\text{SUPR}}$ *singers met* and *The*$_{\text{SUPR}}$ *two singers met*. This exercise can be solved in the Lambda Calculator.

The following exercises compare all three theories of the plural definite article using two models. In addition to the$_{\text{SUM}}$ and the$_{\text{SUPR}}$ introduced above, we include the$_{\iota}$, the straightforward extension of the original iota analysis from Chapter 8 to plural NPs:

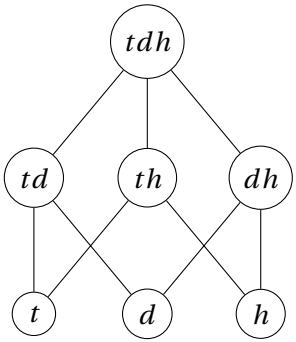
- $\text{the}_{\iota} \rightsquigarrow \lambda P . \iota x . P(x)$
- $\text{the}_{\text{SUM}} \rightsquigarrow \lambda P . \bigoplus P$

[5]The supremum theory was originally proposed by Sharvy (1980), and later, perhaps independently, by Krifka 1986. Champollion & Krifka (2016) write that the Krifka (1986) proposal builds on a suggestion in Montague 1973a, reprinted eight years after Montague's death in 1971 as Montague 1979, but the suggestion there is not quite the supremum theory. Thanks to Ivano Caponigro for help with these scholarly references.

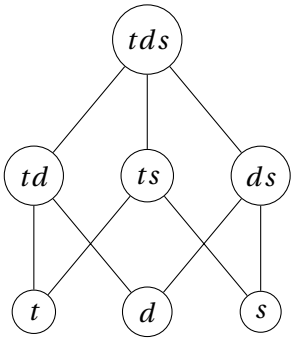
- $\text{the}_{\text{SUPR}} \rightsquigarrow \lambda P.\iota x[P(x) \wedge \forall y[P(y) \rightarrow y \sqsubseteq x]]$

The two models M_1 and M_2 below share the same individuals and sum structure. In M_1, Tom (t), Dick (d), and Harry (h) are all boys. In M_2, Tom (t) and Dick (d) are boys, while Sally (s) is a girl.

M_1 (Tom, Dick, Harry: all boys)



M_2 (Tom, Dick: boys; Sally: girl)



Exercise 13. Compute the denotation of *the boys* under each of the three theories in both M_1 and M_2. Write $\#_e$ if the denotation is undefined. Indicate whether each prediction is correct: *the boys* should denote the plurality of all and only the boys in the model.

M_1:

Theory	Denotation	Correct?
the_ι		
the_{SUM}		
the_{SUPR}		

M_2:

Theory	Denotation	Correct?
the_ι		
the_{SUM}		
the_{SUPR}		

Exercise 14. Compute the denotation of *the two boys* under each theory in both models. (Recall: *two boys* $\rightsquigarrow \lambda x.[\mathsf{card}(x) = 2 \wedge {}^*\mathsf{boy}(x)]$.) Write $\#_e$ if undefined. Indicate whether each prediction is correct: *the two boys* should denote the unique two-boy plurality if one exists, or be undefined if there is no unique such plurality.

M_1:

Theory	Denotation	Correct?
the_ι		
the_{SUM}		
the_{SUPR}		

M_2:

Theory	Denotation	Correct?
the_ι		
the_{SUM}		
the_{SUPR}		

Exercise 15. Combining your results from the previous two exercises, which theory or theories make correct predictions for both *the boys* and *the two boys* across both M_1 and M_2? What does this tell us about the relative merits of the three theories?

9.4 Cumulative readings

So far, we have seen two kinds of predicates that apply to sums: plural nouns like *singers*, and collective predicates like *met*. These are one-place predicates. Sums can also be related by two-place predicates, as in the following sentences:

(42) a. The men in the room are married to the women across the hall. (Kroch, 1974)
b. 600 Dutch firms use 5000 American supercomputers. (adapted from Scha, 1981)
c. Maxine, LaVerne and Patty (between them) own (a total of) four toothbrushes.

Let us take a closer look at the ways the plural entities in these sentences are related. Sentence (42a) is true in a scenario where each of the men in the room is married to one of the women across the hall, and each of the women is married to one of the men. (This might seem to be the only available scenario in which the sentence is true, but this is an effect of Western social/legal norms rather than a linguistic effect. One can easily imagine polygamous societies where other scenarios can be described by this sentence. All that is required for the sentence to be true is that each of the people in the room is married to at least one of the people across from them.) Sentence (42b) (on its relevant reading) is true in a scenario where there are a collection of 600 Dutch firms, and a collection of 5000 American supercomputers, such that each of the firms uses one or more of the supercomputers, and each of the computers is used by one or more of the firms. Sentence (42c) is true in a scenario where Maxine, LaVerne and Patty own toothbrushes in such a way that a total of four toothbrushes are owned. A widespread view is that these scenarios correspond to genuine readings of these sentences, rather than special circumstances under which they are true. These readings are then called *cumulative readings*.

The cumulative reading is distinct from the DISTRIBUTIVE READING, on which each member of the subject group individually stands in the relevant relation to each member of the object group. The distributive reading of sentence (42b) would require each of the 600 firms to individually use 5000 computers — an almost certainly false claim. Cumulative readings are also distinct from collective readings: the collective reading would treat the group of 600 firms as a single entity using the group of 5000 computers as another single entity, with no reference to how the two groups are internally organized.

Cumulative readings, like distributive ones, can be modeled via algebraic closure. The idea is that if Maxine owns toothbrush t_1, LaVerne owns toothbrush t_2, and Patty owns toothbrush t_3 and also toothbrush t_4, then the sum of Maxine, LaVerne and Patty stands in the algebraic closure of the owning relation to the sum of the four toothbrushes. In order to formalize this, we need to generalize the definition of algebraic closure from sets (which correspond to one-place predicates) to n-place relations (which correspond to n-place predicates). We'll focus on the case of a binary (2-place) relation, but the definition can be generalized to relations of arbitrary arity.

(43) **Definition: Sum of a set of pairs**
If R is a set of pairs, then the sum of R, written $\bigoplus R$, is the pair whose first element is the sum of the first elements, and whose second element is the sum of the second elements.

(44) **Definition: Algebraic closure of a set of pairs**
If R is a set of pairs, then the algebraic closure ${}^*R$ of R is the set containing any sum of any nonempty subset of R.

The corresponding symbol in the representation language is defined as follows:

Semantic rule: Relational star operator
If π is an expression of type $\langle e, \langle e, t\rangle\rangle$, then:
$[\![{}^*\pi]\!]^{M,g}$ = that function R such that for all $a, b \in D_e$,
$R(a)(b) = \top$ iff $\langle a, b\rangle \in {}^*\{\langle x, y\rangle \mid [\![\pi]\!]^{M,g}(x)(y) = \top\}$

We can then represent cumulative readings by using the algebraic closure of transitive verbs:

(45) Maxine, LaVerne and Patty own four toothbrushes. $\rightsquigarrow$
$\exists x.\, {}^*\mathsf{toothbrush}(x) \wedge \mathsf{card}(x) = 4 \wedge {}^*\mathsf{own}(\mathsf{mx} \oplus \mathsf{l} \oplus \mathsf{p}, x)$

An example model which verifies formula (45) is the one described above, where Maxine owns toothbrush 1, LaVerne owns toothbrush 2, and Patty owns toothbrushes 3 and 4. The pairs in the relation denoted by "own" are $\langle$Maxine, $t_1\rangle$, $\langle$LaVerne, $t_2\rangle$, $\langle$Patty, $t_3\rangle$ and $\langle$Patty, $t_4\rangle$. The sum of these four pairs is $\langle$Maxine$\oplus$LaVerne$\oplus$Patty, $t_1 \oplus t_2 \oplus t_3 \oplus t_4\rangle$. So there is indeed a value for x that makes the scope of the existential claim true, namely $t_1 \oplus t_2 \oplus t_3 \oplus t_4$.

Exercise 16. Suppose Alice (a) submitted paper p to a conference, and Bob (b) submitted paper q to the same conference, and no other papers were submitted.

1. Is the distributive reading of *Alice and Bob submitted two papers* true in this scenario? Explain.
2. Write a representation of the cumulative reading using ${}^*\mathsf{submit}$.
3. Is the cumulative reading true? Use the definition of the algebraic closure of a set of pairs to verify your answer.

9.5 Summary

This chapter opened with a puzzle about coordination. Once we allow names to be coordinated, their conjunction can serve as the subject of collective predicates — predicates that cannot distribute down to individual members. *John and Sue are a happy couple* cannot mean that John is a happy couple and Sue is a happy couple. What, then, does *John and Sue* denote? The solution we adopted was to enrich the model with a MEREOLOGICAL STRUCTURE: the domain of individuals is closed under a binary sum operation $\oplus$, giving us plural individuals made up of atomic parts. The parthood relation $\sqsubseteq$ provides a natural ordering on this enriched domain, letting us state precisely what it means for one individual to be included in another.

With sums in hand, we turned to the semantics of plural nouns. The plural morpheme *-s* introduces the STAR OPERATOR *, which maps any predicate to its algebraic closure: $^*P$ holds of an individual x exactly when x is a sum of individuals that P holds of. This means $^*\mathsf{singer}$ holds not just of individual singers but of any sum of singers — precisely the extension we want for the English plural *singers*. The star operator is a simple but powerful idea: it makes the singular a special case of the plural rather than a separate semantic category. Numeral modifiers like *two* contribute an independent cardinality condition: $\lambda x.\,\mathsf{card}(x) = 2$, where $\mathsf{card}(x)$ counts the atomic parts of x. Combining a numeral with a plural noun via Predicate Modification yields a predicate like *two singers* that holds of all sums consisting of exactly two singers.

Plural definite descriptions raised a further question: which theory of the definite article correctly predicts the presuppositions of *the singers*? We compared three candidates — the ι operator, the sum theory, and the supremum theory — and found that only the supremum handles all cases correctly. The ι operator fails for plurals entirely, since $\iota x.\,{}^*\mathsf{boy}(x)$ is always undefined when there is more than one boy. The sum theory handles basic plurals but

overgenerates for modified plurals like *the two boys*: the sum of all boys is not the same individual as the sum of all two-boy pluralities. Only the supremum — the smallest individual that includes all the relevant atoms — delivers the right presupposition in every case.

Finally, we extended algebraic closure from one-place predicates to binary relations. The RELATIONAL STAR OPERATOR $^*\pi$ applied to a binary predicate π returns a relation that holds between sums a and b whenever the original predicate holds of their parts in a matching way. This captures CUMULATIVE READINGS: *Maxine, LaVerne, and Patty own four toothbrushes* is true not because any one of them owns all four, but because the sum of their individual ownership pairs falls within the algebraic closure of the *own* relation. The same formal tool — algebraic closure — thus unifies the semantics of the plural morpheme, collective predication, and cumulative transitive constructions within a single framework built on mereology.

The formal definitions and lexical entries for L_* are collected below.

9.5.1 Logic syntax

The syntax of L_* is defined as in three-valued type logic (L_∂), with two extensions. First, we add n (natural numbers) to the set of basic types; card is a non-logical constant of type $\langle e, n\rangle$ that counts the atomic parts of an individual. Second, we add new syntactic rules introducing the part-of relation, the binary sum operator, the generalized sum operator, and the unary and binary algebraic closure operators.

1. **Parthood**
   If α and β are terms, i.e. expressions of type e, then $\alpha \sqsubseteq \beta$ is an expression of type t.

2. **Binary sum**

If α and β are terms, i.e. expressions of type e, then $\alpha \oplus \beta$ is an expression of type e.

3. **Sum of a set**
   If π is an expression of type $\langle e, t\rangle$, then $\bigoplus \pi$ is an expression of type e.

4. **Closure of a predicate**
   If π is an expression of type $\langle e, t\rangle$, then $*\pi$ is an expression of type $\langle e, t\rangle$.

5. **Closure of a relation**
   If π is an expression of type $\langle e, \langle e, t\rangle\rangle$, then $*\pi$ is an expression of type $\langle e, \langle e, t\rangle\rangle$.

We write $\alpha \sqsubset \beta$ 'alpha is a proper part of beta' as an abbreviation for $\alpha \sqsubseteq \beta \wedge \alpha \neq \beta$ 'alpha is a part of beta and alpha is distinct from beta'. We also write $\mathsf{atom}(\alpha)$ as an abbreviation for $\neg \exists y[y \sqsubset \alpha]$ 'alpha has no proper parts'.

9.5.2 Logic semantics

A model for L_* based on D is a 4-tuple

$$\langle (D_\tau)_{\tau \in \mathcal{T}}, (D_\tau)^+_{\tau \in \mathcal{T}}, I, \sqsubseteq \rangle$$

where:

- $(D_\tau)_{\tau \in \mathcal{T}}$ is a standard frame based on D
- $(D_\tau)^+_{\tau \in \mathcal{T}}$ is an augmented frame based on D
- for every type $\tau \in \mathcal{T}$, I assigns to every non-logical constant of type τ an object from the domain D^+_τ.
- $\sqsubseteq$ is a partial order over D_e.

The last bullet is the new; the other three are carried over from Chapter 8.

We have defined a number of notions based on the part-of relation:

- **Generalized sum**
  Relative to a given model M determining the part relation $\sqsubseteq$, given a set $S \subseteq D_e$,

  $\bigoplus S \stackrel{\text{def}}{=}$ the individual k in D_e such that:
  (i) for all $x \in S$, $x \sqsubseteq k$;
  (ii) there is no $k' \sqsubset k$ such that for all $x \in S$, $x \sqsubseteq k'$.

- **Algebraic closure of a set of individuals**
  ${}^*S \stackrel{\text{def}}{=} \{x \mid \text{there exists } S' \text{ s.t. } S' \neq \varnothing \text{ and } S' \subseteq S \text{ and } x = \bigoplus S'\}$

- **Sum of a set of pairs**
  If R is a set of pairs, then the sum of R, written $\bigoplus R$, is the pair whose first element is the sum of the first elements, and whose second element is the sum of the second elements.

- **Algebraic closure of a set of pairs**
  If R is a set of pairs, then the algebraic closure ${}^*R$ of R is the set containing any sum of any nonempty subset of R.

Based on these definitions, we have laid out the following semantic rules for the new symbols of the representation language.

- **Parthood**
  If α and β are expressions of type e and M determines the individual-part relation $\sqsubseteq$:

$$[\![\alpha \sqsubseteq \beta]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g} \sqsubseteq [\![\beta]\!]^{M,g} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

- **Binary sum**
  If α and β are expressions of type e and M determines the

part-of relation $\sqsubseteq$:

$$[\![\alpha \oplus \beta]\!]^{M,g} = [\![\alpha]\!]^{M,g} \oplus [\![\beta]\!]^{M,g}.$$

- **Generalized sum**
  If π is an expression of type $\langle e, t\rangle$, then:
  $[\![\bigoplus \pi]\!]^{M,g} = \bigoplus\{x \mid [\![\pi]\!]^{M,g}(x) = \top\}$

- **Star operator**
  If π is an expression of type $\langle e, t\rangle$, then:
  $[\![^* \pi]\!]^{M,g} =$ that function f such that
  for all $x \in D_e$, $f(x) = \top$ iff $x \in {}^*\{y \mid [\![\pi]\!]^{M,g}(y) = \top\}$

- **Relational star operator**
  If π is an expression of type $\langle e, \langle e, t\rangle\rangle$, then:
  $[\![^* \pi]\!]^{M,g} =$ that function R such that for all $a, b \in D_e$,
  $R(a)(b) = \top$ iff $\langle a, b\rangle \in {}^*\{\langle x, y\rangle \mid [\![\pi]\!]^{M,g}(x)(y) = \top\}$

9.5.3 English syntax

Syntax rules. We add the following rule for the plural:

$$\text{N} \quad \rightarrow \quad \text{N PL}$$

Schemas introduced so far (where *XP* ranges over any phrasal category):

XP	→	Neg *XP*
S	→	SNeg S
XP	→	*XP* &*XP*
&*XP*	→	& *XP*

Lexicon. Lexical items are associated with syntactic categories as follows:

D:	$\varnothing_D$
A:	*two, three* etc.

&:	*and, or*
PL:	*-s*
V:	*met, own*

9.5.4 Translations

Words of English will be translated into L_* as follows. We keep the same composition rules as in previous chapters.

Type $\langle e, t\rangle$:

1. *smokes* $\rightsquigarrow \lambda x.{}^*\mathsf{smokes}(x)$
2. *drinks* $\rightsquigarrow \lambda x.{}^*\mathsf{drinks}(x)$
3. *two* $\rightsquigarrow \lambda x.\mathsf{card}(x) = 2$

Type $\langle e, \langle e, t\rangle\rangle$:

1. *caught* $\rightsquigarrow \lambda y \lambda x.{}^*\mathsf{catch}(y)(x)$
2. *ate* $\rightsquigarrow \lambda y \lambda x.{}^*\mathsf{eat}(y)(x)$
3. *own* $\rightsquigarrow \lambda y \lambda x.{}^*\mathsf{own}(y)(x)$

Type e:

1. *John* $\rightsquigarrow \mathsf{j}$
2. *Sue* $\rightsquigarrow \mathsf{s}$
3. *Mary* $\rightsquigarrow \mathsf{m}$
4. *Maxine* $\rightsquigarrow \mathsf{mx}$
5. *LaVerne* $\rightsquigarrow \mathsf{l}$
6. *Patty* $\rightsquigarrow \mathsf{p}$

Type $\langle t, \langle t, t\rangle\rangle$:

1. $and_S \rightsquigarrow \lambda q \lambda p.p \wedge q$

2. $or_S \rightsquigarrow \lambda q \lambda p . p \vee q$

Type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$:

1. *is, a* $\rightsquigarrow \lambda P . P$

Type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,\langle e,t\rangle\rangle\rangle$:

1. $and_{VP} \rightsquigarrow \lambda P' \lambda P \lambda x . P(x) \wedge P'(x)$
2. $or_{VP} \rightsquigarrow \lambda P' \lambda P \lambda x . P(x) \vee P'(x)$

Type $\langle\langle e,\langle e,t\rangle\rangle,\langle\langle e,\langle e,t\rangle\rangle,\langle e,\langle e,t\rangle\rangle\rangle\rangle$:

1. $and_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . R(y)(x) \wedge R'(y)(x)$
2. $or_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . R(y)(x) \vee R'(y)(x)$

Type $\langle\langle e,t\rangle,e\rangle$:

1. *the* $\rightsquigarrow \lambda P . \iota z[P(z) \wedge \forall x[P(x) \rightarrow x \sqsubseteq z]]$

Type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$:

1. *-s* $\rightsquigarrow \lambda P . {}^*P$

Type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$:

1. $\varnothing_D \rightsquigarrow \lambda P \lambda Q . \exists x . [P(x) \wedge Q(x)]$

10 Event semantics

10.1 Introduction

Recall that one of the advantages of translating natural language into logic is that it helps us account for certain entailment relations between natural language sentences. Suppose that whenever a sentence A is true, a sentence B is also true. If the translation of A logically entails that of B, then we have an explanation for this entailment. Take the following sentences:

(1) a. John smokes and Mary drinks.
b. ∴ John smokes.

This argument is captured by the following logical entailment:

(2) a. $[\mathsf{smokes(j)} \wedge \mathsf{drinks(m)}]$
b. $\mathsf{smokes(j)}$

In every model where (2a) is true, (2b) is also true.

This pattern of inference – a longer sentence entails a shorter one – also shows up in other places. Adverbial modification is one example.

(3) a. Jones buttered the toast slowly.
b. ∴ Jones buttered the toast.

Here is how we would translate (3a) given the previous chap-

ters (we are treating *the toast* as if it were a constant rather than a definite description, but nothing will hinge on this):

(4) $\mathsf{butter(j,t)}$

If this representation is correct, (3a) is about only two entities: Jones and the toast. Which entity does *slowly* describe in (3a)? Is it perhaps Jones who is slow? Then we might translate that sentence as follows:

(5) $[\mathsf{butter(j,t)} \wedge \mathsf{slow(j)}]$

Since (5) logically entails (4), we have an account of the entailment from (3a) to (3b). But there is a problem. If we represent (3a) as (5), clearly we ought to translate (6a) as (6b), by analogy.

(6) a. Jones buttered the bagel quickly.
b. $[\mathsf{butter(j,b)} \wedge \mathsf{quick(j)}]$

But then, in any model where (5) and (6b) are both true, the following will also be true as a matter of logical consequence!

(7) $[\mathsf{slow(j)} \wedge \mathsf{quick(j)}]$

Unless we want to countenance the possibility that Jones is both slow and quick at the same time, our account clearly has a problem.

One might think that the slowness is not a property of Jones, but of whatever *Jones buttered the toast* denotes. This would lead us to a translation of (3a) like this:

(8) $\mathsf{slow(butter(j,t))}$

In the system we have been developing so far, there is a problem with this idea too. The denotation of the subformula $\mathsf{butter(j,t)}$ is a truth value, and correspondingly, its type is t. Since the constant slow in (8) is predicated of that subformula, it has to denote a function whose input type is t. And if the entire formula (8) is

to denote a truth value, the output type of slow is t as well, and its type as a whole must be $\langle t, t\rangle$. But there are only two truth values (setting aside the undefined truth value), so there are only four functions of that type: the identity function, negation, the function that maps both truth values to T, and the function that maps both truth values to F. None of these functions captures the truth conditions of slow.

So slow cannot have the type $\langle t, t\rangle$. What if it has the type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$, and applies to the VP *buttered the toast* and then to *Jones*?

(9) $\mathsf{slow}(\lambda x.\mathsf{butter}(x, \mathsf{t}))(\mathsf{j})$

The problem with this approach is that it does not explain the pattern of inference shown in (3). Depending on what slow denotes in any given model, the entailment from *Jones buttered the toast slowly* to *Jones buttered the toast* may or may not hold. We can remedy this by stipulating a meaning postulate to the effect that whenever a property P that holds of an individual x is modified by slow, it still holds of x:

(10) $\forall P \forall x.\mathsf{slow}(P)(x) \rightarrow P(x)$

But this is no more than a stopgap measure. It is analogous to what we did in Chapter 7 when we gave intersective adjectives, such as *reasonable* and *vegetarian*, the type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$ and accounted for their intersectivity by a meaning postulate. In that chapter, we also saw that we can give intersective adjectives a simpler type instead, and remove the need for a meaning postulate. The solution we are about to adopt is analogous.

In an influential paper, Davidson (1967) suggested that it is not Jones but the action – or, as we will say, the *event* – of buttering the toast that is slow in (3a). On Davidson's view, events are taken to be concrete entities with locations in space and in time, and natural language provides means to express information about them, refer to them, etc. Although not all sentences

that are about events necessarily provide explicit clues to that effect, some do. For example, the subjects in these two sentences arguably have an event as their referent (Parsons, 1990):

(11) a. Jones' buttering of the toast was artful.
b. It happened slowly.

So let us assume that in (3a) it is the event of buttering the toast that is slow, and in (6a) it is the event of buttering the bagel that is quick, rather than Jones himself. The two sentences, then, are not only talking about Jones and the things he is buttering but also about the buttering events. According to Davidson (1967), the correct logical representations for (3a) and (6a) are not (5) and (6b) but rather something like the following:

(12) a. $\exists e.[\mathsf{butter}(\mathsf{j},\mathsf{t},e) \wedge \mathsf{slow}(e)]$
b. $\exists e.[\mathsf{butter}(\mathsf{j},\mathsf{b},e) \wedge \mathsf{quick}(e)]$

Here, the variable e stands for an event, and the existential quantifier that binds it ranges only over events, and not over individuals. Correspondingly, we introduce a new basic type for events, alongside the type e of individuals and the type t of truth values. Since the letter e is already taken, it is common to use v for the type of events, as we will do here. (In some papers, the type of events is also written ϵ.)

A sentence like (3b) would then be represented as:

(13) $\exists e.\mathsf{butter}(\mathsf{j},\mathsf{t},e)$

There is a logical entailment from (12a) to (13), as desired. But unlike before, the conjunction of (12a) and (12b) no longer entails that something is both slow and quick at the same time, since the two formulas could (and typically will) be true in virtue of different events.

Adverbs like *quickly* and *slowly* are not the only phenomena in natural language that have been given an event semantic treat-

ment – far from it. Here are a few other examples.

Prepositional adjuncts. Adjuncts like *in the kitchen* and *at noon* can be dropped from ordinary true sentences without affecting their truth value. Moreover, when a sentence has multiple adverbs and adjuncts then one or more can be dropped. In these respects, they behave just like the adverbs *quickly* and *slowly* that we have already seen:

(14) a. Jones buttered the toast slowly in the kitchen at noon.
b. ∴ Jones buttered the toast slowly in the kitchen.
c. ∴ Jones buttered the toast slowly.
d. ∴ Jones buttered the toast.

Event semantics provides a straightforward account of these entailment patterns:

(15) a. $\exists e.\mathsf{butter}(\mathsf{j},\mathsf{t},e) \wedge \mathsf{slow}(e) \wedge \mathsf{loc}(e,\mathsf{k}) \wedge \mathsf{time}(e,\mathsf{noon})$
b. $\exists e.\mathsf{butter}(\mathsf{j},\mathsf{t},e) \wedge \mathsf{slow}(e) \wedge \mathsf{loc}(e,\mathsf{k})$
c. $\exists e.\mathsf{butter}(\mathsf{j},\mathsf{t},e) \wedge \mathsf{slow}(e)$
d. $\exists e.\mathsf{butter}(\mathsf{j},\mathsf{t},e)$

Direct perception and causation reports. Since events are concrete entities with a location in spacetime, it stands to reason that we can see and hear them, and that they can be involved in causal relations. This idea can be exploited to give semantics of direct perception reports and causation reports (Higginbotham, 1983):

(16) a. John saw Mary leave.
b. ∴ Mary left.

(17) a. John made Mary leave.
b. ∴ Mary left.

(18) a. $\exists e \exists e'.\mathsf{see}(\mathsf{j},e',e) \wedge \mathsf{leave}(\mathsf{m},e')$
b. $\exists e'.\mathsf{leave}(\mathsf{m},e')$

(19) a. $\exists e \exists e'.\mathsf{cause}(\mathsf{j},e',e) \wedge \mathsf{leave}(\mathsf{m},e')$

b. $\exists e'.\mathsf{leave}(\mathsf{m}, e')$

Here, e is the event of John seeing or causing something, and e' is the event seen or caused by John—that is, the event of Mary leaving.

The relation between adjectives and adverbs. If adverbs ascribe properties to events, it is plausible to assume that the same is true of adjectives that are derivationally related to these adverbs (Parsons, 1990):

(20) a. Brutus stabbed Caesar violently.
b. ∴ Something violent happened.

(21) a. $\exists e.\mathsf{stab}(\mathsf{b},\mathsf{c},e) \wedge \mathsf{violent}(e)$
b. $\exists e.\mathsf{violent}(e)$

10.1.1 The Neo-Davidsonian turn

As we have seen, Davidson equipped verbs with an additional event argument. Later authors, however, have taken the event to be the only argument of the verb (e.g. Castañeda, 1967; Parsons, 1990). The relationship between this event and syntactic arguments of the verb is then expressed by a smallish number of semantic relations with names like AGENT, THEME, INSTRUMENT, and BENEFICIARY. These relations represent ways entities take part in events and are generally called THEMATIC ROLES. The first occurrence of thematic roles is as the six kāraka relations in the Aṣṭādhyāyī, a precise formal grammar of Classical Sanskrit created nearly 2500 years ago by Dakṣīputra Pāṇini, arguably the first descriptive linguist. In modern times, two influential works are Gruber (1965) and Jackendoff (1972). This came to be known as "Neo-Davidsonian" event semantics. Thematic roles describe semantic relations between events and their participants in terms that generalize across many verbs. For example, the agent initiates and carries out the event; the theme undergoes the event and does not have control

over the way it occurs; the instrument is manipulated by an agent and is used to perform an intentional act; the beneficiary is potentially advantaged or disadvantaged by the event; and so on. Additional thematic roles that specify the location of an event in space and time are often proposed. For events of perception, one finds the roles STIMULUS (the cause) and EXPERIENCER (the patient that is aware of the event undergone), and for motion events, the roles SOURCE and GOAL for the initial and final points. The label PATIENT is sometimes used interchangeably with THEME, and we will follow this convention here. Sometimes a distinction is drawn between the two, in that patients undergo a change of state as a result of an event but themes do not. There is no consensus on the full inventory of thematic roles, but role lists of a large number of English verbs have been compiled in Levin (1993) and Kipper-Schuler (2005). An ISO standard for thematic roles was established under the label ISO 24617-4:2014.

On the Neo-Davidsonian view, *Jones buttered the toast* might be represented as follows:

(22) $\exists e.\mathsf{butter}(e) \wedge \mathsf{agent}(e,\mathsf{j}) \wedge \mathsf{theme}(e,\mathsf{t})$

In Neo-Davidsonian event semantics, there is no fundamental semantic distinction between syntactic arguments such as the subject and object of a verb, and syntactic adjuncts such as adverbs and prepositional phrases. For example, in the following representation of *Jones buttered the toast with a knife*, the conjunct that represents the prepositional phrase is essentially parallel to those conjuncts that represent *Jones* and *the toast*. (For simplicity, we represent *a knife* as if it were a constant. Just like in the case of *the toast*, this is not essential.)

(23) $\exists e.\mathsf{butter}(e) \wedge \mathsf{agent}(e,\mathsf{j}) \wedge \mathsf{theme}(e,\mathsf{t}) \wedge \mathsf{instr}(e,\mathsf{k})$

The idea that there is no fundamental semantic distinction between syntactic arguments and adjuncts might not be immediately clear. In what way is the prepositional phrase *with a knife*

parallel to the argument *the toast*? The following pair can make this clearer.

(24) a. Mary loaded the truck with the hay.
b. Mary loaded the hay onto the truck.

Setting aside slight semantic differences between these two sentences, their common semantic core can be expressed in the following way: there is a loading event whose agent is Mary, whose goal (or location, on some accounts) is the truck, and whose theme is the hay. This is expressed in the following translation:

(25) $\exists e.\mathsf{load}(e) \wedge \mathsf{agent}(e,\mathsf{m}) \wedge \mathsf{goal}(e,\mathsf{t}) \wedge \mathsf{theme}(e,\mathsf{h})$

The argument *the truck* in (24a) parallels the adjunct *onto the truck* in (24b), and the adjunct *with the hay* in (24a) parallels the argument *the hay* in (24b).

One consequence of the lack of a semantic distinction between arguments and adjuncts is that on the Neo-Davidsonian view, sentences with too many or too few arguments are ungrammatical but not semantically deviant. The following sentences can all be assigned coherent event semantic translations, unlike in eventless or classical Davidsonian semantics, where the number of semantic arguments of a verb is fixed.

(26) a. John ate.
b. John ate the fish.
c. John dined.
d. *John dined the fish.
e. *John devoured.
f. John devoured the fish.

This aspect of Neo-Davidsonian event semantics has been justified in terms of the lack of any semantic distinction between verbs with different subcategorization frames such as *eat*, *dine*, and *devour* that could explain why the first is optionally intransi-

tive, the second is obligatorily so, and the third obligatorily transitive. Whatever distinction there is between them must arguably instead be attributed to syntax.

One of the advantages of the Neo-Davidsonian view is that it allows us to capture semantic entailment relations between different syntactic subcategorization frames of the same verb, such as causatives and their intransitive counterparts (Parsons, 1990):

(27) a. Mary opened the door.
b. ∴ The door opened.

(28) a. $\exists e.\mathsf{open}(e) \wedge \mathsf{agent}(e,\mathsf{m}) \wedge \mathsf{theme}(e,\mathsf{d})$
b. $\exists e.\mathsf{open}(e) \wedge \mathsf{theme}(e,\mathsf{d})$

The Neo-Davidsonian approach raises important questions, many of which have been answered in different ways in the semantic literature. Do semantic roles have syntactic counterparts? If so, how should we think of them? For example, presumably the thematic role of *Mary* in (29a) – perhaps beneficiary – matches the one of *Mary* in (29b).

(29) a. Jane gave the ball to Mary.
b. Jane gave Mary the ball.

We might think of this role as the denotation of *to* in (29a), but in (29b) there is no corresponding word we can point to. One common perspective on thematic roles in generative syntax is that when no preposition is around, they are assigned by (usually silent) functional heads projected in the syntax, often called *theta roles*. For example, a "little v" head is often assumed to relate verbs to their external arguments, which are usually their agents; here the little v head would be the theta role and the agent relation the thematic role (Chomsky, 1995). As another example, the preposition *with* often serves as the theta role of the thematic role *instrument*. We follow the textbook Carnie (2013) in using the term *thematic role* for the semantic relation, and the term *theta role* for its syn-

tactic counterparts; however, some authors use these terms interchangeably.

Another question is whether each verbal argument (perhaps with the exception of dummy subjects as in *It's raining*) corresponds to exactly one role, or whether the subject of a verb like *fall* is both the agent and the theme (or patient or experiencer) of the event (Parsons, 1990). Relatedly, it is often assumed that each event has at most one agent, at most one theme, and so on. (If the domain of individuals includes sums of individuals, as in Chapter 9, it is common to assume that the domain of events includes sums of events as well. The agent of a sum of events is then taken to be the sum of their agents, and similarly for other thematic roles.) This view, often called the *unique role requirement* or *thematic uniqueness*, is widely accepted in semantics (Carlson, 1984; Parsons, 1990; Landman, 2000). Thematic uniqueness has the effect that thematic roles can be represented as partial functions. This is often reflected in the notation, as in (30).

(30) $\exists e.\,\mathsf{butter}(e) \wedge \mathsf{agent}(e) = \mathsf{j} \wedge \mathsf{theme}(e) = \mathsf{t}$

A differing, less common view is based on the intuition that one can touch a man and his shoulder in the same event (Krifka, 1992). In this example, one could argue that there is a single touching event that stands in the theme relation both to the man and to his shoulder.

10.2 Composition in Neo-Davidsonian event semantics

Building Neo-Davidsonian semantics into our fragment requires us to decide how events, event quantifiers, and thematic roles enter the compositional process. There is currently no universally accepted way to settle the question. A common approach is that verbs and verbal projections (such as VPs and IPs) denote predicates of events and are intersected with their arguments and ad-

juncts, until an existential quantifier is inserted at the end and binds the event variable (Carlson, 1984; Parsons, 1990, 1995). We call this the EVENT-PREDICATE APPROACH.

10.2.1 Verbs as predicates of events

Denotations of verbs. Under the event-predicate approach, verbs denote predicates of events:

(31) a. *bark* $\rightsquigarrow \lambda e.\mathsf{bark}(e)$
b. *butter* $\rightsquigarrow \lambda e.\mathsf{butter}(e)$
c. ...

These lexical entries conform with the Neo-Davidsonian view in that they do not contain any variables for the arguments of the verb. So now the question is how to link up the arguments of the verb with the events described by the verbs.

Theta role heads. One way to achieve argument-linking is to allow each noun phrase a way to "sprout" a theta role head θ.

(32) **Syntax**
DP $\rightarrow$ θ DP

We then write lexical entries that map these heads to suitable roles:

(33) **Lexicon**
θ: AGENT, THEME, ...

At this point, we would normally need to make sure that the right syntactic argument gets mapped to the right thematic roles. For example, the subject is typically, but not exclusively, mapped to the agent role. Operations such as passivization change the order in which arguments get mapped to thematic roles. This is what theories of argument structure are about (e.g. Wunderlich, 2012). We will ignore this problem here and simply assume that each θ head gets mapped to the "right" role.

Next, we give lexical entries to these theta role heads. The thematic role constants agent and theme have type $\langle v, e\rangle$, mapping events to individuals. The theta role heads denote these constants directly:

(34) a. AGENT $\rightsquigarrow$ agent, type $\langle v, e\rangle$
b. THEME $\rightsquigarrow$ theme, type $\langle v, e\rangle$
c. ...

To combine with a noun phrase and yield an event predicate of type $\langle v, t\rangle$, a theta role head must be type-shifted. When the noun phrase denotes an individual (type e), the Individual θ-Participant rule applies; when it denotes a predicate (type $\langle e, t\rangle$), the Predicate θ-Participant rule applies. In trees, $\Uparrow_{\text{IND}}$ and $\Uparrow_{\text{PRED}}$ indicate which rule has applied.

Type-Shifting Rule. Individual θ-Participant (IND)
If $\alpha \rightsquigarrow \theta$ of type $\langle v, e\rangle$, then:

$$\alpha \rightsquigarrow \lambda x \lambda e.\, \theta(e) = x$$

as well (type $\langle e, \langle v, t\rangle\rangle$).

Type-Shifting Rule. Predicate θ-Participant (PRED)
If $\alpha \rightsquigarrow \theta$ of type $\langle v, e\rangle$, then:

$$\alpha \rightsquigarrow \lambda P \lambda e.\, P(\theta(e))$$

as well (type $\langle\langle e, t\rangle, \langle v, t\rangle\rangle$).

Existential Closure rule. Finally, we introduce an operation that existentially binds the event variable at the sentence level. We can handle this operation as a type-shifting rule. Here, and in what

follows, v stands for the type of events, so $\langle v, t \rangle$ is the type of an event predicate.

Type-Shifting Rule. Existential Closure ($\Uparrow_{\exists}$)
If $\alpha \rightsquigarrow \alpha'$, where α' is of category $\langle v, t \rangle$, then:

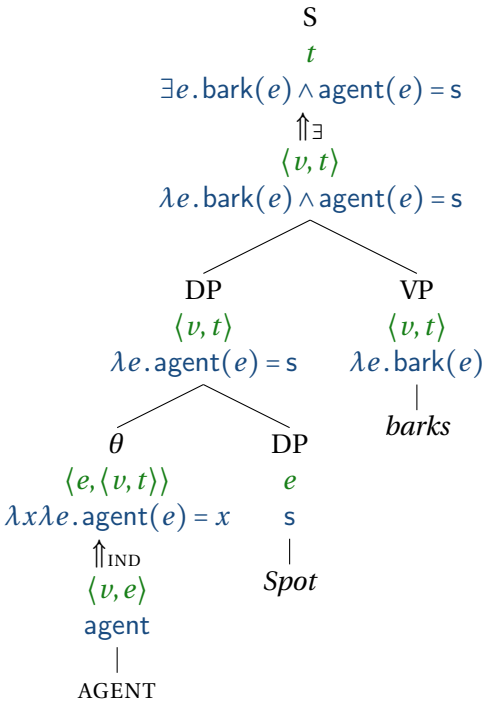
$$\alpha \rightsquigarrow \exists e . \alpha'(e)$$

as well (as long as e does not occur in α'; in that case, use a different variable of the same type).

(Note: we write $\Uparrow_{\exists}$ with the standard mathematical symbol for existential quantification. This is easy to confuse with $\Uparrow_{\text{EX}}$, which uses the small-caps abbreviation EX and means the same thing. We consistently use $\Uparrow_{\exists}$ in this chapter.)

A sample derivation that shows two of the elements we have introduced is shown in (35). The subject and the verb phrase both denote predicates of events, and combine via Predicate Modification. The resulting event predicate is mapped to a truth value by the Existential Closure type-shifting rule.

(35)



In subsequent trees, we abbreviate the theta role leaf as $\text{AGENT}\Uparrow_{\text{IND}}$ (or $\text{THEME}\Uparrow_{\text{IND}}$, $\text{AGENT}\Uparrow_{\text{PRED}}$, etc.), omitting the base type $\langle v, e\rangle$ and base semantics below the arrow.

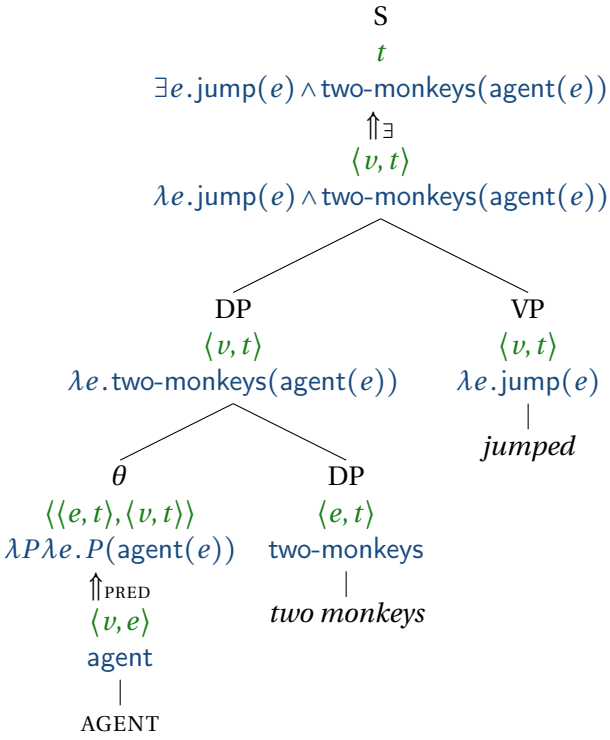
The existential closure type-shifting rule applies at the root of the tree. Since both VP and S have the same type, one might wonder what prevents it from applying at VP. In that case, the type of VP would be t and there would be no way for the subject to combine with it. As long as the syntax requires that a subject is present, this derivation will not be interpretable.

The Predicate θ-Participant rule applies when a DP argument denotes a predicate of type $\langle e, t\rangle$ rather than an individual of type e. This arises with predicative noun phrases such as *two monkeys*, which denote predicates of type $\langle e, t\rangle$ (see Chapter 9). The shifted entries are:

(36) a. $\text{AGENT}\Uparrow_{\text{PRED}} \rightsquigarrow \lambda P \lambda e.\, P(\text{agent}(e))$, type $\langle\langle e,t\rangle,\langle v,t\rangle\rangle$
b. $\text{THEME}\Uparrow_{\text{PRED}} \rightsquigarrow \lambda P \lambda e.\, P(\text{theme}(e))$, type $\langle\langle e,t\rangle,\langle v,t\rangle\rangle$

To illustrate, consider *Two monkeys jumped*. Using the lexical entries for *two* and the plural morpheme *-s* from Chapter 9, we can derive a denotation for *two monkeys*; we abbreviate this as two-monkeys, of type $\langle e,t\rangle$. Since *two monkeys* denotes a predicate rather than an individual, $\text{AGENT}\Uparrow_{\text{PRED}}$ applies, as shown in (37).

(37)



Exercise 1. Compositionally derive meanings for the following two sentences, using the supremum analysis of the definite article from Chapter 9, and the type-shifters for the theta role heads introduced above.

(i) *The two monkeys jumped*

(ii) *The kids saw two monkeys* (use $\text{AGENT}\Uparrow_{\text{IND}}$ for the subject, $\text{THEME}\Uparrow_{\text{PRED}}$ for the object)

Adverbs. Let us now add the adjunct *slowly* to our fragment. This adverb is quite free in terms of where it can occur in the sentence: before the sentence, between subject and VP, and at the end of the sentence. This is captured in the following rules:

(38) **Syntax**

S	→	AdvP S
VP	→	AdvP VP
VP	→	VP AdvP
AdvP	→	Adv

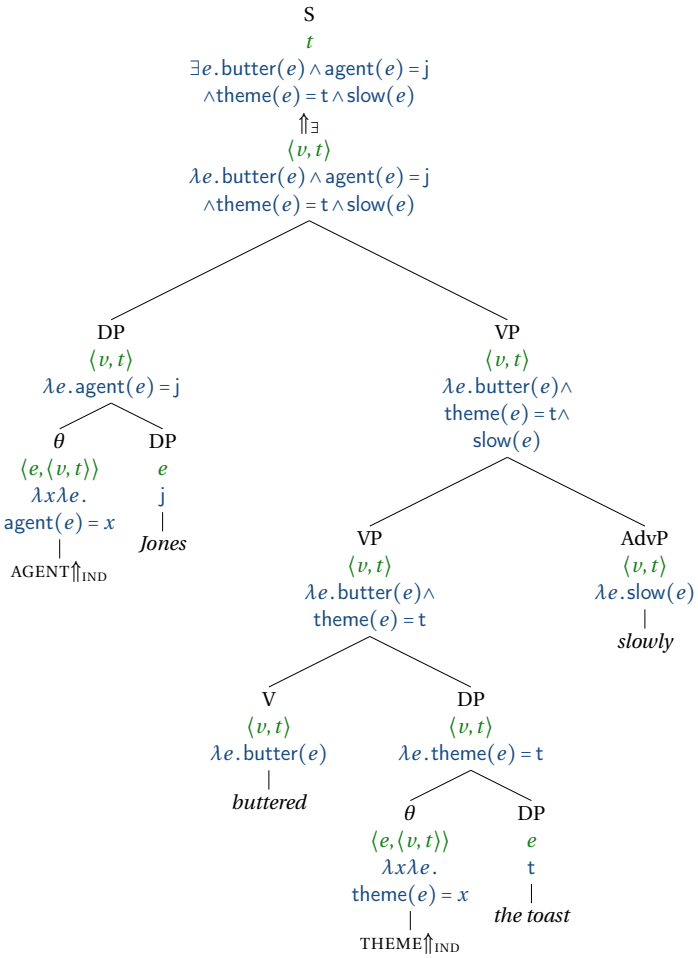
(39) **Lexicon**
Adv: slowly

As we have seen above, *slowly* is interpreted as an event predicate. Its lexical entry is therefore very simple:

(40) a. *slowly* $\rightsquigarrow \lambda e.\mathsf{slow}(e)$

The tree in (41) shows the application of *slowly*. Like the subject and object, it is a predicate of type $\langle v, t\rangle$ and it combines with its sister node via Predicate Modification:

(41)



In the derivation in (41), syntactic arguments do not change the type of the verbal projections they attach to. This is a hallmark of Neo-Davidsonian event semantics. The object maps a predicate of type $\langle v, t\rangle$ (the V) to another one that is also of type $\langle v, t\rangle$ (the VP). The subject maps a predicate of type $\langle v, t\rangle$ (the VP) to another one that is also of type $\langle v, t\rangle$ (the S). This is very different from what we have seen in previous chapters, where V, VP and S all

had different types (namely, $\langle e, \langle e, t\rangle\rangle$, $\langle e, t\rangle$, and t respectively). In Neo-Davidsonian semantics, syntactic arguments are semantically indistinguishable (as far as types are concerned) from adjuncts, which map a VP of a certain type (here, $\langle v, t\rangle$) to another VP of the same type and which do not change the type of the VP.

10.3 Negation in event semantics

Let us now consider how to analyse a sentence with negation:

(42) Spot didn't bark.

Observe that (42) only has the reading in (43b), and lacks the reading in (44b).

(43) a. $\neg[\exists e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}]$
b. "There is no barking event that is done by Spot"

(44) a. $\exists e.\neg[\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}]$
b. "There is an event that is not a barking by Spot"

The reading in (44b), if it were available, would be almost trivially true: any event that isn't a barking event by Spot would verify it. Empirically, it seems that negation always takes scope over the event quantifier. The question is how this result is obtained compositionally by the grammar. We consider four hypotheses.

Hypothesis 1: Negation interpreted at clause level. Let us begin with a hypothesis that encodes the empirical observation that negation outscopes the existential quantifier over the event quite directly. On this hypothesis, *not* takes scope over the entire clause, including the subject, so that it applies to *Spot did bark* as a whole rather than just to the VP. It maps the resulting event predicate to a proposition of type t. We use V as a variable of type $\langle v, t\rangle$:

(45) $\mathit{not} \rightsquigarrow \lambda V.\neg\exists e.V(e)$, type $\langle\langle v, t\rangle, t\rangle$

Applying this to the event predicate $\lambda e.\,\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}$ yields $\neg\exists e.\,\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}$, which is (43b).

This hypothesis does require negation to be interpreted quite high—not merely immediately above the subject, but above any modifier that itself takes scope over the subject. Consider:

(46) In the garden, Spot didn't bark.

Here the PP *in the garden* takes scope over the subject, meaning it has attached to the clause above the subject's position. For Hypothesis 1 to derive the correct truth conditions, *not* must be interpreted above *in the garden* as well.

So far, this hypothesis appears to be viable. But would it be possible to construct a theory on which negation is interpreted in situ? Hypotheses 2–4 illustrate three possibilities.

Hypothesis 2: Negation as set complement. Recall that under the event-predicate approach, verbs and their projections denote sets of events. For example, *bark* denotes the set of all barking events, and so does the VP *bark*. If negation is interpreted at the VP level, then negation must map this set to another set of events. What could that set be? Here is one attempt:

(47) $\textit{not} \rightsquigarrow \lambda V \lambda e.\,\neg V(e)$, type $\langle\langle v, t\rangle, \langle v, t\rangle\rangle$

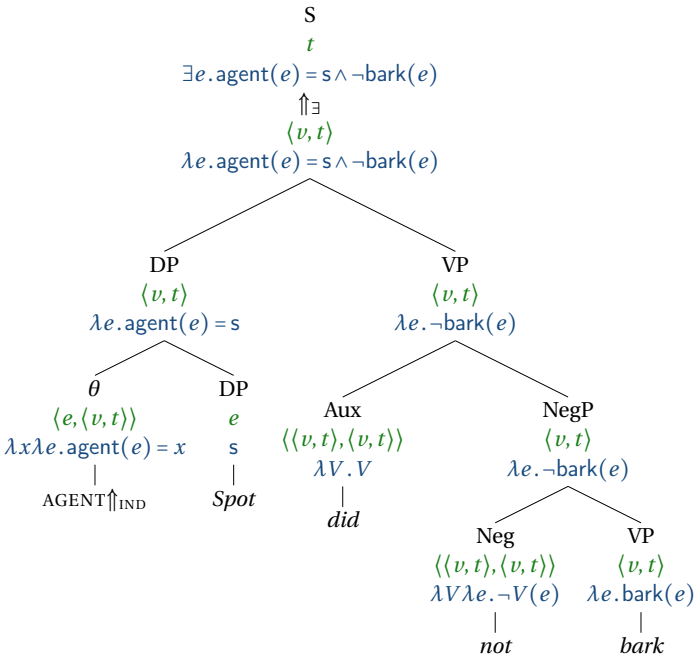
But this is a disastrous denotation. It says that *not* applies to a set of events and maps it to its complement—the complement of the set of barking events. The subject then combines via the thematic role head, and existential closure applies at the sentence level. The result is:

(48) $\exists e.\,\mathsf{agent}(e) = \mathsf{s} \wedge \neg\mathsf{bark}(e)$
"There is an event whose agent is Spot that is not a barking event."

This derivation is shown in (49). What we want is (43b). But For-

mula (48) is true just in case Spot did anything at all instead of or in addition to barking. The problem runs deeper than the faulty entry in (47): it is conceptually not clear what set of events should be denoted by *not bark*, nor what it would take for an event to be a member of this set. Hypothesis 2 is ruled out.

(49)



Hypothesis 3: Negative events. Another strategy for interpreting negation in situ would be to expand our inventory of events to include "negative events". In some cases, it is intuitively clear what a negative event should be: for example, a negative staying event can be thought of as a leaving event and vice versa. While it is not so clear what a negative barking event is, some semanticists have tried to clarify their status (Bernard & Champollion, 2018). We set Hypothesis 3 aside rather than rule it out; working out a full theory of negative events would take us beyond the scope of

this chapter.

Hypothesis 4: Negation under the event-quantifier approach. This hypothesis requires a radically different analysis of verbs and verb phrases. We begin by presenting this new analysis, called the EVENT-QUANTIFIER APPROACH (Champollion, 2014), and then come back to negation.

On the event-quantifier approach, verbs come equipped with their own event quantifiers. Verbs no longer denote event predicates but rather generalized existential quantifiers over events. We equip each verb with a variable V of type $\langle v, t\rangle$, a variable over sets of events. This variable stands for the future of the derivation, that is, for the semantic contributions of any constituents (arguments and adjuncts) that are about to combine with the verb. Variables that stand for the future of the derivation are known as CONTINUATION VARIABLES (Barker & Shan, 2014). We include the event quantifier into the lexical entry for each verb and give it scope over the variable V and thereby over any other quantifiers that might be contributed over the future course of the derivation. The new representations for verbs are as follows:

(50) a. $\textit{bark} \rightsquigarrow \lambda V \exists e.\mathsf{bark}(e) \wedge V(e)$
b. $\textit{butter} \rightsquigarrow \lambda V \exists e.\mathsf{butter}(e) \wedge V(e)$
c. ...

Our grammar will continue to map verbal projections (verbs, VPs and Ss) to the same type. But this type is no longer $\langle v, t\rangle$ but $\langle\langle v, t\rangle, t\rangle$. We use $\mathcal{Q}$ as a variable over event quantifiers, i.e., expressions of type $\langle\langle v, t\rangle, t\rangle$.

Using the event-quantifier approach, we no longer rely on predicate modification to combine verbs with their arguments. Instead, we use function application to combine syntactic arguments with verbal projections. The base entries for AGENT and THEME remain type $\langle v, e\rangle$ (as introduced in Section 10.2.1). The individual-selecting shifted entries take the following form under this ap-

proach:

(51) a. $\text{AGENT}\Uparrow_{\text{IND+}} \rightsquigarrow \lambda x \lambda \mathcal{Q} \lambda V . \mathcal{Q}(\lambda e . \text{agent}(e) = x \wedge V(e))$
b. $\text{THEME}\Uparrow_{\text{IND+}} \rightsquigarrow \lambda x \lambda \mathcal{Q} \lambda V . \mathcal{Q}(\lambda e . \text{theme}(e) = x \wedge V(e))$
c. ...

Here, $\Uparrow_{\text{IND+}}$ is a type-shifting operation that takes an unadulterated thematic role head and converts it into a function that expects an event quantifier as its first argument. This type shift and the corresponding $\Uparrow_{\text{PRED+}}$ shift are defined as follows:

Type-Shifting Rule. Individual θ-Participant (IND+)
If $\alpha \rightsquigarrow \theta$ of type $\langle v, e \rangle$, then:

$$\alpha \rightsquigarrow \lambda x \lambda \mathcal{Q} \lambda V . \mathcal{Q}(\lambda e . \theta(e) = x \wedge V(e))$$

as well (type $\langle e, \langle \langle \langle v, t \rangle, t \rangle, \langle \langle v, t \rangle, t \rangle \rangle \rangle$).

Type-Shifting Rule. Predicate θ-Participant (PRED+)
If $\alpha \rightsquigarrow \theta$ of type $\langle v, e \rangle$, then:

$$\alpha \rightsquigarrow \lambda P \lambda \mathcal{Q} \lambda V . \mathcal{Q}(\lambda e . P(\theta(e)) \wedge V(e))$$

as well (type $\langle \langle e, t \rangle, \langle \langle \langle v, t \rangle, t \rangle, \langle \langle v, t \rangle, t \rangle \rangle \rangle$).

If the root of the tree is of type $\langle \langle v, t \rangle, t \rangle$, we need to map it to a truth value. In a simple case such as *Spot barks*, the root will be true of any set of events V so long as V contains (possibly among other things) an event that satisfies the relevant event predicate. Whether this is true can be checked by testing whether the set of all events whatsoever, $\lambda e . \text{true}$, contains such an event:

(52) a. $\lambda V \exists e [\text{bark}(e) \wedge \text{agent}(e) = \text{s} \wedge V(e)](\lambda e . \text{true})$

b. $\exists e[\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s} \wedge (\lambda e.\mathsf{true})(e)]$ (β)
c. $\exists e[\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s} \wedge \mathsf{true}]$ (β)
d. $\exists e[\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}]$

After the type-$\langle\langle v, t\rangle, t\rangle$ expression at the root of the tree applies to $\lambda e.\mathsf{true}$, the result is of type t, as desired. To formalize this idea, we introduce the type-shifting rule of Quantifier Closure:

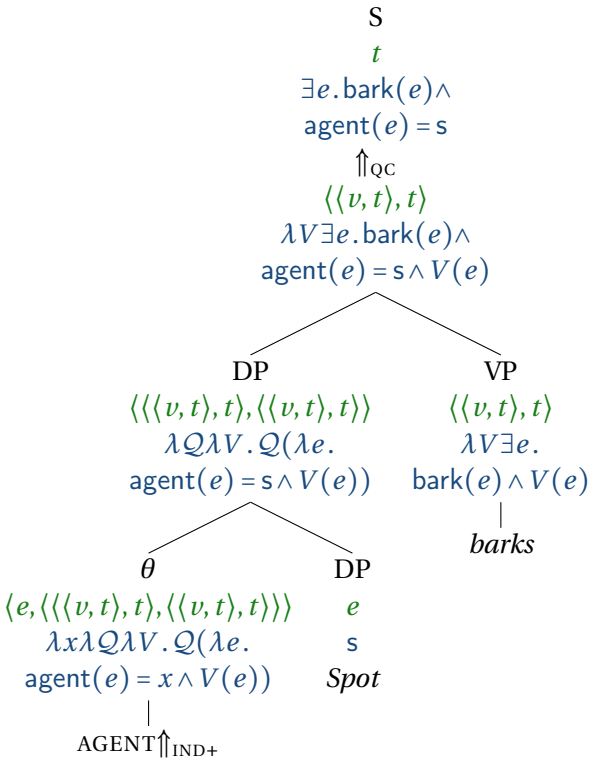
Type-Shifting Rule. Quantifier Closure (QC)
If $\alpha \rightsquigarrow \alpha'$, where α' is of category $\langle\langle v, t\rangle, t\rangle$, then:

$$\alpha \rightsquigarrow \alpha'(\lambda e.\mathsf{true})$$

as well.

A full derivation for *Spot barked* on the event-quantifier approach is given in (53):

(53)

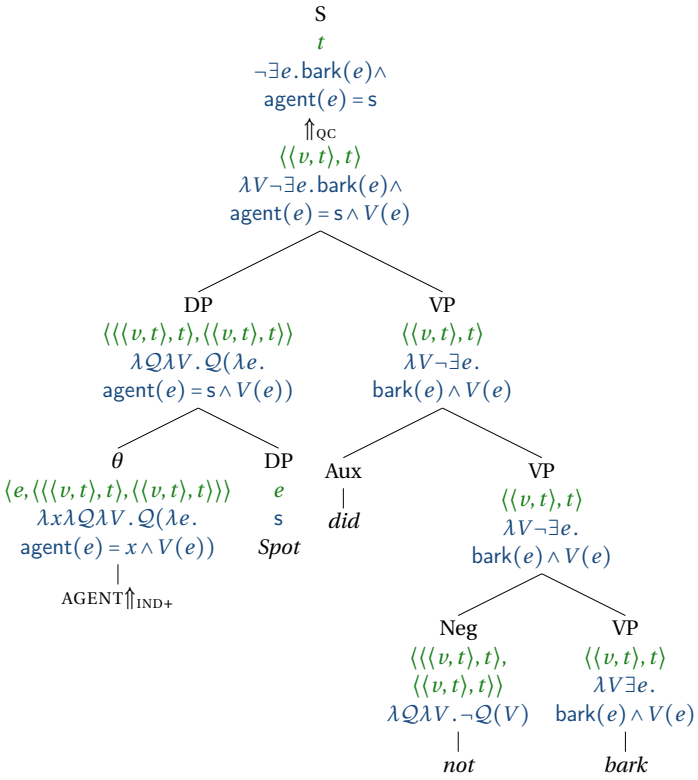


Negation, then, would work as follows. On this approach, *not bark* can be given a straightforward denotation—the set of sets of events that do not contain any barking events—by giving *not* the following entry:

(54) *not* $\rightsquigarrow \lambda\mathcal{Q}\lambda V.\neg\mathcal{Q}(V)$, type $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle$

The resulting truth conditions are the desired ones in (43b). The derivation for (42) is shown in (55).

(55)



(The auxiliary *did* carries tense information but is treated as semantically vacuous here; the VP it dominates inherits the semantics of its sister VP directly.)

So the event-quantifier approach yields a satisfactory in-situ account of negation in event semantics. It turns out that the event-quantifier approach to verb semantics yields better results than the event-predicate approach when it comes to both coordination and quantification, as we will see in the forthcoming sections.

Exercise 2. Using the event-quantifier approach, compositionally derive the meaning of the following sentence. Show the type and

semantic value at each node.

(i) *Brutus didn't stab Caesar*

10.4 Conjunction in event semantics

In Chapter 9, we have seen that many uses of *and* can be subsumed under a general schema, discussed by Partee & Rooth (1983) among others. This schema is repeated here:

(56) $\text{and}_{\langle\tau,\langle\tau,\tau\rangle\rangle}$

$$\rightsquigarrow \begin{cases} \lambda q \lambda p.\, p \wedge q & \text{if } \tau = t \\ \lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1}.\, \alpha'(X(Z))(Y(Z)) & \text{if } \tau = \langle\sigma_1,\sigma_2\rangle \end{cases}$$

where $\text{and}_{\langle\sigma_2,\langle\sigma_2,\sigma_2\rangle\rangle} \rightsquigarrow \alpha'$.

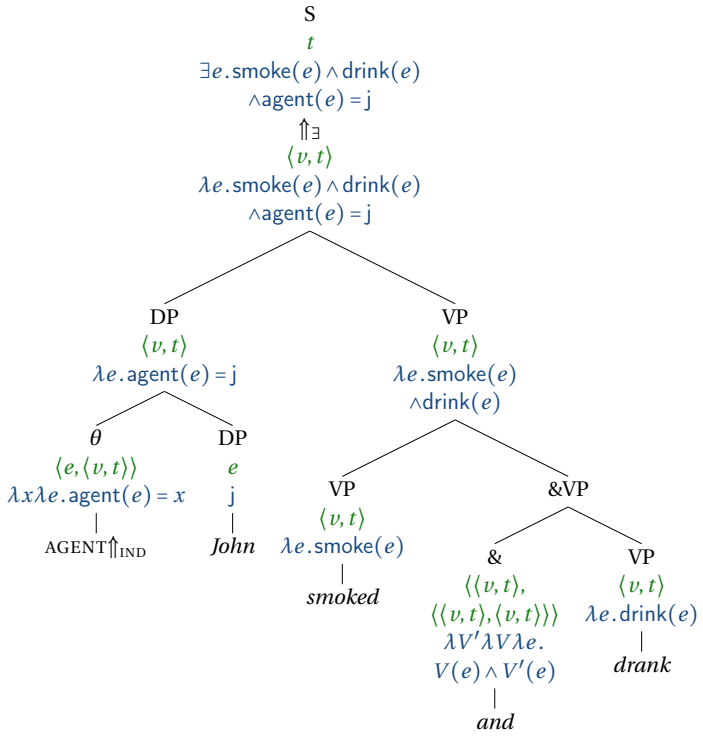
We now consider what this schema amounts to in the case of VP-modifying *and*, as in *John smoked and drank*.

Conjunction under the event-predicate approach. Under the event-predicate approach, VPs are of type $\tau = \langle v, t\rangle$. Applying rule (56) yields the following entry for *and*:

(57) $\text{and}_{\text{VP}} \rightsquigarrow \lambda V' \lambda V \lambda e.\, V(e) \wedge V'(e)$

A derivation for *John smoked and drank* using this entry is shown in (58):

(58)



This yields the following translation for *John smoked and drank*:

(59) $\exists e.\mathsf{smoke}(e) \wedge \mathsf{drink}(e) \wedge \mathsf{agent}(e) = \mathsf{j}$

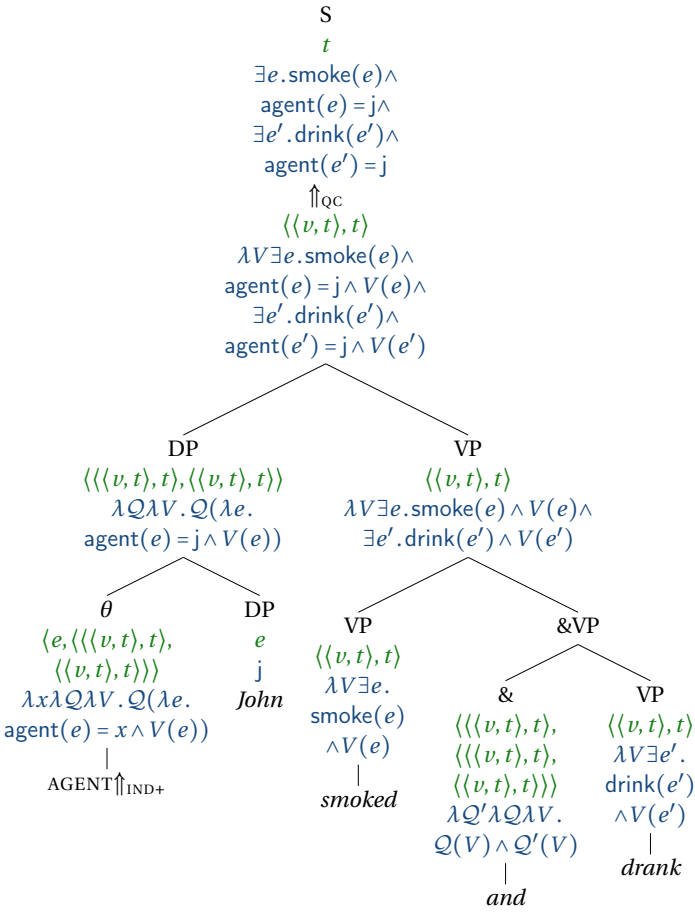
Now, (59) cannot be the right representation of *John smoked and drank*. If this sentence is true, for all we know he might have smoked slowly and drunk quickly. In (59) there is only one event for two such contradictory adverbs to modify, and we would end up with the same kind of problem we already encountered earlier in connection with *Jones buttered the toast slowly and buttered the bagel quickly*. Part of the point of introducing events was to avoid having to attribute contradictory properties to the same entity. The entry in (59) sends us right back to square one.

Conjunction under the event-quantifier approach. Under the event-quantifier approach, VPs are of type $\tau = \langle\langle v, t\rangle, t\rangle$. Applying rule (56) yields the following entry for *and*:

(60) $\text{and}_{\text{VP}} \rightsquigarrow \lambda\mathcal{Q}'\lambda\mathcal{Q}\lambda V\,.\,\mathcal{Q}(V) \wedge \mathcal{Q}'(V)$

A derivation for *John smoked and drank* using this entry is shown in (61):

(61)



This yields the following translation for *John smoked and drank*:

(62) $\exists e.\mathsf{smoke}(e) \wedge \mathsf{agent}(e) = \mathsf{j} \wedge$
$\exists e'.\mathsf{drink}(e') \wedge \mathsf{agent}(e') = \mathsf{j}$

We fare much better with (62): it provides us with the two events we need to avoid the problem of contradictory adverb modification.

Exercise 3. Add *slowly* and *quickly* to the tree in (61) and show how the resulting formula avoids the attribution of contradictory properties to the same event.

Exercise 4. Show how rule (56) leads to the entries in (57) and (60).

A sum-based approach to coordination. The Partee & Rooth schema is not compatible with the event-predicate approach, but this does not mean that conjunction cannot be represented under that approach. We can still formulate an entry for VP-level conjunction that is compatible with event predicates. This is similar to DP-level conjunction, where in Chapter 9 we have encountered both schema-based and non-schema-based entries.

Exercise 5. Formulate an entry for VP-level conjunction that is compatible with the event-predicate approach. Hint: use sums of events. Make sure it predicts the right truth conditions for *John smoked slowly and drank quickly.* Assume that for any theta role θ, $\theta(e \oplus e') = \theta(e) \oplus \theta(e')$.

10.5 Quantification in event semantics

Let us now consider what happens when we bring in quantifiers like *every cat* and *no dog*. Since the event variable is bound by a silent existential quantifier, we might expect that in this case too any overt quantifiers in the sentence can take scope either over or under it. But this is not the case. Rather, the event quantifier always takes scope *below* anything else in the sentence. For example, sentence (63), read with neutral intonation, is not ambiguous. Its only reading corresponds to (64b), where the event quantifier takes low scope. As for (65b), that is not a possible reading of the sentence.

(63) No dog barks.

(64) a. $\neg[\exists x.\mathsf{dog}(x) \wedge \exists e[\mathsf{bark}(e) \wedge \mathsf{agent}(e) = x]]$
b. "There is no barking event that is done by a dog"

(65) a. $\exists e.\neg[\mathsf{bark}(e) \wedge \exists x[\mathsf{dog}(x) \wedge \mathsf{agent}(e) = x]]$
b. "There is an event that is not a barking by a dog"

Exercise 6. How can you tell that (65b) is not a possible reading of sentence (63)?

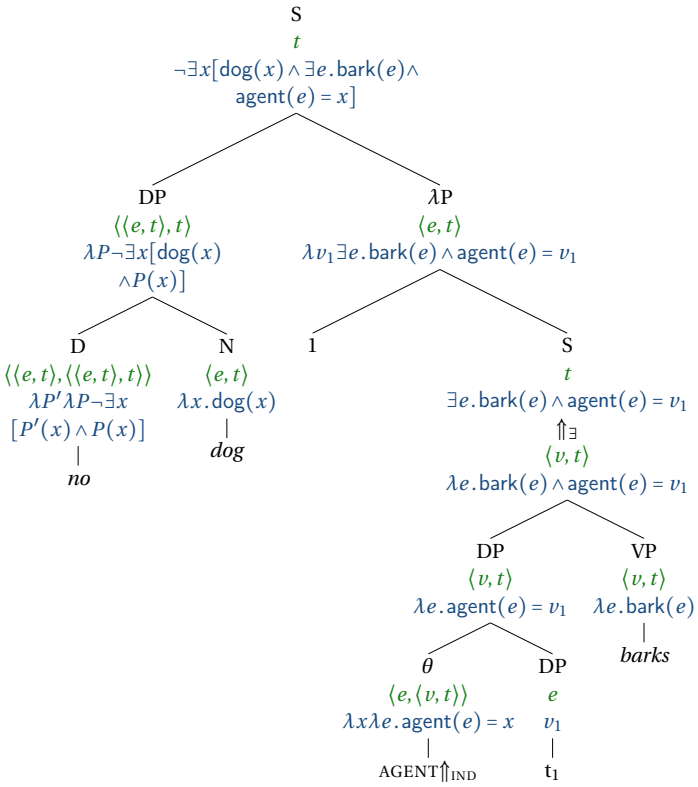
10.5.1 Verbs as event predicates

In the standard (non-event) setting, RAISE-O and RAISE-S offer an alternative to quantifier raising: they lift a verb to accept a generalized quantifier directly, allowing NPs to be interpreted in situ. In the event-semantic setting, however, these operations do not behave as expected.

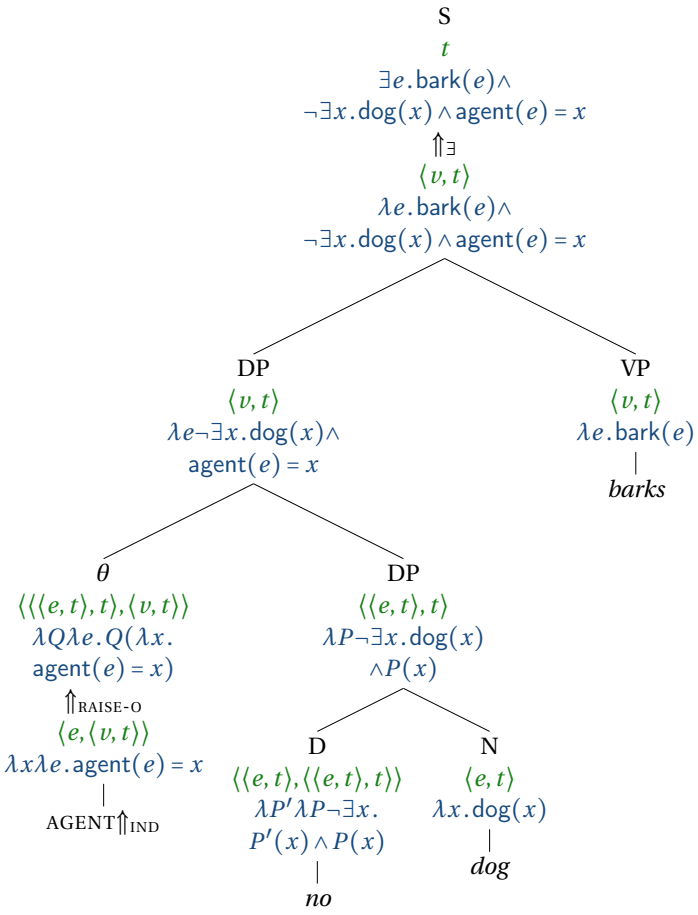
To see the problem in the event-predicate approach, consider *No dog barks*. After IND, the theta role head AGENT has type $\langle e, \langle v, t\rangle\rangle$, which satisfies the type condition for RAISE-O. Applying RAISE-O

does yield a derivation, shown in (67). But the result is the unavailable reading (65b): because Existential Closure introduces $\exists e$ at sentence level, the event existential ends up *inside* the scope of *no*. The correct reading (64b) requires quantifier raising instead, moving *no dog* above sentence level and hence above Existential Closure, as shown in (66).

(66)



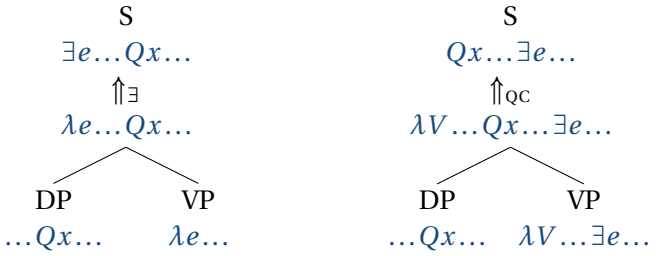
(67)



The interim conclusion is that the event-predicate approach commits us to a quantifier-raising treatment of quantificational NPs: applying RAISE-O is formally possible but yields the wrong scope, because the event existential introduced at sentence level ends up inside the scope of the NP quantifier rather than above it.

10.5.2 Verbs as event quantifiers

The event-quantifier approach has a different character. Because each verb carries its own event quantifier — $\exists e$ is built into the lexical entry rather than introduced by a sentential rule — the NP quantifier already sits above the event existential in the surface structure, as illustrated in Figure 10.1b. Quantifier raising is therefore not needed to get the right scope.



(a) $\exists e$ introduced at S (b) $\exists e$ introduced by verb

Figure 10.1: Comparison of two approaches to event semantics. Note the position of the existential quantifier over events in each subfigure.

More interestingly, the event-quantifier approach is also compatible with a type-shifting treatment of quantificational NPs — something the event-predicate approach cannot offer. The standard RAISE-O operation does not directly apply in this setting: after IND+, the theta role head has type $\langle e, \langle \langle \langle v, t \rangle, t \rangle, \langle \langle v, t \rangle, t \rangle \rangle \rangle$, which is more complex than the $\langle e, \langle a, t \rangle \rangle$ form that RAISE-O requires. However, there exists a type shift in the spirit of RAISE-O that does apply: it replaces the individual argument of type e with a generalized quantifier of type $\langle \langle e, t \rangle, t \rangle$, passing the remaining arguments through. Applied to the theta role head, this shift yields the correct reading for *No dog barks* without any quantifier raising, since $\exists e$ is already inside the verb and thus already in the scope of the NP quantifier. The precise formulation of this shift

and its relationship to Hendriks's general type-lifting schema are worked out in Champollion (2014).

The upshot is that the event-quantifier approach has a theoretical advantage when it comes to quantification: it is compatible with a directly compositional account of NPs, in which all NPs are interpreted in situ. The event-predicate approach is not, because RAISE-O gives the wrong scope in that setting. This is a point in favour of the event-quantifier approach, particularly for the analysis of languages where scope is not flexible and in-situ interpretation is expected (Huang, 1981; Champollion, 2014).

10.6 Aktionsart and telicity

The compositional analyses developed in this chapter treat events as objects related to their participants through thematic roles. But events differ not only in who participates in them: they also differ in their *temporal constitution*, for example, in whether or not they have a natural culmination point. The study of these distinctions is called AKTIONSART (German: 'manner of action', also known as LEXICAL ASPECT), and it turns out to connect the semantics of events directly to the mereological architecture introduced in Chapter 9.

Vendler classes. Vendler (1957) proposed a four-way classification of English predicates based on their inherent temporal properties:

(68) a. *States*: *know*, *love*, *contain* — stative; hold uniformly over an extended interval
b. *Activities*: *run*, *swim*, *push a cart* — dynamic processes with no inherent culmination point
c. *Accomplishments*: *eat an apple*, *build a house*, *drink a glass of wine* — processes with a natural culmination point

d. *Achievements*: *reach the summit, die, recognize someone* — instantaneous transitions to a new state

One important dimension is TELICITY: roughly, whether the predicate has a natural culmination point or not. Accomplishments and achievements are TELIC; states and activities are ATELIC.

The for/in test. A key diagnostic for telicity is the distribution of *for* and *in* adverbials. An adverbial of the form *in X time* measures the interval up to the culmination of an event; it is felicitous only with telic predicates. Adverbials of the form *for X time* are felicitous with atelic predicates.

(69) a. Jones ran for an hour.
b. *Jones ran in an hour.
c. Jones ate an apple in ten minutes.
d. *Jones ate an apple for ten minutes.

The felicity of (69)a and (relative) infelicity of (69)b show that the bare VP *run* is atelic. The felicity of (69)c and (relative) infelicity of (69)d show that *eat an apple* is telic.

Object-determined telicity. For many verbs, telicity is not a fixed property of the verb alone but depends on the object NP. The same verb can head either a telic or an atelic VP depending on its complement:

(70) a. Jones ate apples for an hour.
b. Jones ate an apple in ten minutes.
c. Jones ate three apples in half an hour.

In (70)a, the bare plural *apples* yields an atelic VP; *for an hour* is felicitous. In (70)b, the singular indefinite *an apple* yields a telic VP; *in ten minutes* is felicitous. In (70)c, the numeral NP *three apples* likewise yields a telic VP. This phenomenon, in which the grammatical object affects telicity, holds across a wide range of tran-

sitive verbs: *drink*, *read*, *build*, and many others (Verkuyl, 1972; Krifka, 1989, 1998).

The pattern is not limited to the count/mass distinction. Generally, it tracks the BOUNDEDNESS of the theme: informally, whether the object NP specifies a delimited quantity. A bounded NP (*an apple*, *three apples*), makes the VP telic; an NP denoting an unbounded quantity (bare plural *apples*, bare mass *wine*) makes it atelic.

The incremental theme analysis. Why should the object NP determine the telicity of the VP? The key intuition, developed formally by Krifka (1989, 1998), is that some events are *measured out* by their themes. Eating an apple is complete precisely when the apple has been consumed: the endpoint of the event corresponds to the extent of the object. Such predicates are called INCREMENTAL THEME predicates.

This can be made precise using the part-of relation from Chapter 9. Events, like individuals, stand in part-of relations: the first five minutes of a ten-minute apple-eating are a *subevent* of the whole eating. Writing $e' \sqsubset_v e$ for the event-level part relation (the analogue of $\sqsubseteq$ for individuals), an incremental theme predicate P satisfies two conditions (Krifka, 1998):

(71) Mapping to Objects
$$\forall e \forall e' \big[P(e) \wedge e' \sqsubset_v e \rightarrow \mathsf{theme}(e') \sqsubseteq \mathsf{theme}(e) \big]$$

(72) Mapping to Events
$$\forall e \forall x \big[P(e) \wedge x \sqsubseteq \mathsf{theme}(e) \rightarrow \exists e' \big[e' \sqsubset_v e \wedge \mathsf{theme}(e') = x \big] \big]$$

(71) says that every subevent of a P-event has a theme that is part of the whole event's theme. (72) says that every part of the theme corresponds to some subevent. Together they establish a bijective homomorphism between the mereological structure of events and the mereological structure of their themes.

This account explains object-determined telicity. The bare plural *apples* denotes the cumulative predicate $^*\mathsf{apple}$, which holds

of any sum of apples however large. With no upper bound on the sorts of objects that can serve as theme, the homomorphism imposes no upper bound on the sorts of events that the VP describes. This property makes the VP atelic under Krifka's definition of (a)telicity.

On the other hand, the singular indefinite *an apple* denotes a predicate true of individual atomic apples. Two apples together would not satisfy the description *an apple*, in contrast to the *apples* case. Similarly, the numeral NP *three apples* denotes a predicate true of sums with cardinality three: $\lambda x.{}^{*}\mathsf{apple}(x) \wedge \mathsf{card}(x) = 3$ (see Chapter 9). Combining a group of three apples with another apple or three would render the predicate *three apples* inapplicable.

The mereological analysis thus reveals a deep connection between noun phrase semantics and event semantics, first noted by Bach (1986): both have part structure. So just as there is a part relation $\sqsubseteq$ that holds between individuals, there is also a part relation that holds between events. We refer the reader to Krifka (1989, 1998) and Champollion (2017) for a fuller exposition of this connection.

10.7 Toy fragment

The following is an updated version of the toy fragment of Chapter 8, extended with the mereological additions from Chapter 9 and the event-semantic additions from this chapter. The common fragment below is shared by both the event-predicate approach (Section 10.2.1) and the event-quantifier approach (Section 10.5.2). Extensions A and B then present the approach-specific type-shifting rules and lexical entries for verbs.

Representation language.
This chapter extends L_{*} (Chapter 9) with a new basic type v for events. All syntactic and semantic rules of L_{*} carry over unchanged.

Type conventions.

- Variables of type e: x, y, z, and indexed variants x_i
- Variables of type $\langle e, t\rangle$: P, Q
- Variables of type $\langle\langle e, t\rangle, t\rangle$: Z
- Variables of type $\langle e, \langle e, t\rangle\rangle$: R
- Variables of type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: f
- Variables of type t: p, q
- Variables of type v: e, e'
- Variables of type $\langle v, t\rangle$: V
- Variables of type $\langle\langle v, t\rangle, t\rangle$: $\mathcal{Q}$
- Constants of type e: a, b, c, j (Jones), s (Spot), t (the toast)
- Constants of type $\langle e, t\rangle$: smiled, laughed, smokes, drinks, kind, swedish, happy, proud, sailor, pirate, singer, drummer, musician, person, male
- Constants of type $\langle e, \langle e, t\rangle\rangle$: loves, likes, hugged, with, sister, teacher
- Constants of type $\langle v, t\rangle$: bark, butter, jump, see, smoke, drink, slow, quick
- Constants of type $\langle v, e\rangle$: agent, theme

Variables of types not listed above carry an explicit type subscript (e.g., x_a for a variable of schema type a). When multiple variable symbols appear in the same formula, they are understood to be distinct from one another and from any indexed variant (x_i) in the same formula.

Syntax.

S	→	DP VP
S	→	SNeg S
VP	→	V (DP|AP|PP)
AP	→	A (PP)
DP	→	D (NP)
DP	→	DP[GEN] NP
DP[GEN]	→	DP Poss
NP	→	N (PP)
NP	→	A NP
NP	→	NP CP
CP	→	DP_i [WH] S
PP	→	P DP
DP	→	θ DP
S	→	AdvP S
VP	→	VP AdvP
AdvP	→	Adv

Schemas (where *XP* ranges over any phrasal category):

XP	→	Neg *XP*
XP	→	*XP* &*XP*
&*XP*	→	& *XP*

Syntactic transformations. Both syntactic transformations from the fragment in Chapter 7 (relative clause formation, QR) carry over.

Composition rules. All composition rules from the Chapter 8 toy fragment carry over, with Predicate Modification extended to cover type $\langle v, t\rangle$ in addition to $\langle e, t\rangle$.

Lexical entries (common to both approaches).

All type-shifting operations and lexical entries from the Chapter 8 toy fragment carry over. Verb entries are replaced by the approach-specific entries in Extensions A and B below. New entries introduced in this chapter:

θ (thematic role heads)

- AGENT $\rightsquigarrow$ agent
  (type $\langle v, e \rangle$)
- THEME $\rightsquigarrow$ theme
  (type $\langle v, e \rangle$)

Adv

- *slowly* $\rightsquigarrow$ $\lambda e.\mathsf{slow}(e)$
  (type $\langle v, t \rangle$)
- *quickly* $\rightsquigarrow$ $\lambda e.\mathsf{quick}(e)$
  (type $\langle v, t \rangle$)

Extension A: Event-predicate approach.

Verbs denote predicates of events (type $\langle v, t \rangle$). Thematic role heads are type-shifted to link noun phrase arguments to the event; the event variable is bound at the sentence level by Existential Closure. Verbal projections at all levels (V, VP, S) share type $\langle v, t \rangle$, and verb arguments combine with the verb via Predicate Modification.

Type-shifting operations

- **Individual θ-Participant** (IND)
  If $\alpha \rightsquigarrow \theta$ of type $\langle v, e \rangle$, then α also has a translation of type $\langle e, \langle v, t \rangle \rangle$:
  $$\lambda x \lambda e.\theta(e) = x$$

- **Predicate θ-Participant** (PRED)
  If $\alpha \rightsquigarrow \theta$ of type $\langle v, e\rangle$, then α also has a translation of type $\langle\langle e, t\rangle, \langle v, t\rangle\rangle$:
  $$\lambda P \lambda e . P(\theta(e))$$

- **Existential Closure** ($\Uparrow_{\exists}$)
  If $\alpha \rightsquigarrow \alpha'$ of type $\langle v, t\rangle$, then α also has a translation of type t:
  $$\exists e . \alpha'(e)$$
  (with e chosen so as not to occur free in α').

Lexical entries

V

bark, *jump* and other intransitive verbs follow $W \rightsquigarrow \lambda e . W(e)$; *butter*, *see*/*saw* and other transitive verbs follow $W \rightsquigarrow \lambda e . W(e)$ as well, since thematic role heads supply the argument links.

- *bark* $\rightsquigarrow \lambda e . \mathsf{bark}(e)$
- *jump* $\rightsquigarrow \lambda e . \mathsf{jump}(e)$
- *butter* $\rightsquigarrow \lambda e . \mathsf{butter}(e)$
- *see*/*saw* $\rightsquigarrow \lambda e . \mathsf{see}(e)$

Neg

- *not* $\rightsquigarrow \lambda V . \neg \exists e . V(e)$
  (type $\langle\langle v, t\rangle, t\rangle$)

Extension B: Event-quantifier approach.

Verbs are generalized existential quantifiers over events (type $\langle\langle v,t\rangle,t\rangle$), equipped with a continuation variable V that stands for the semantic contributions of future arguments and adjuncts. Verb arguments combine via Function Application. Quantifier Closure maps the root expression of type $\langle\langle v,t\rangle,t\rangle$ to a truth value.

Type-shifting operations

- **Individual θ-Participant** (IND+)
  If $\alpha \rightsquigarrow \theta$ of type $\langle v,e\rangle$, then α also has a translation of type $\langle e,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$:

$$\lambda x\lambda\mathcal{Q}\lambda V.\mathcal{Q}(\lambda e.\theta(e)=x\wedge V(e))$$

- **Predicate θ-Participant** (PRED+)
  If $\alpha \rightsquigarrow \theta$ of type $\langle v,e\rangle$, then α also has a translation of type $\langle\langle e,t\rangle,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$:

$$\lambda P\lambda\mathcal{Q}\lambda V.\mathcal{Q}(\lambda e.P(\theta(e))\wedge V(e))$$

- **Quantifier Closure** (QC)
  If $\alpha \rightsquigarrow \alpha'$ of type $\langle\langle v,t\rangle,t\rangle$, then α also has a translation of type t:

$$\alpha'(\lambda e.\mathsf{true})$$

Lexical entries

V

bark, jump, butter, see/saw and other verbs follow $W \rightsquigarrow \lambda V.\exists e.W(e)\wedge V(e)$ (type $\langle\langle v,t\rangle,t\rangle$).

- *bark* $\rightsquigarrow \lambda V.\exists e.\mathsf{bark}(e)\wedge V(e)$
- *jump* $\rightsquigarrow \lambda V.\exists e.\mathsf{jump}(e)\wedge V(e)$

- *butter* $\rightsquigarrow \lambda V.\exists e.\mathsf{butter}(e) \wedge V(e)$
- *see*/*saw* $\rightsquigarrow \lambda V.\exists e.\mathsf{see}(e) \wedge V(e)$

Neg

- *not* $\rightsquigarrow \lambda \mathcal{Q} \lambda V.\neg \mathcal{Q}(V)$
  (type $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle$)

11 Tense and aspect

11.1 Introduction

We turn, now, finally, to the analysis of tense. Tense and aspect, rather, as the topics are inextricably intertwined, as we shall see. The chapter begins with three desiderata for a theory of tense. The subsequent sections present a theory that is aimed at satisfying them.

11.1.1 Some desiderata for a theory of tense

11.1.1.1 The deictic nature of tense

The first point that we would like to make is that tense is DEICTIC, or in other words INDEXICAL. By this, we mean that the temporal reference of a tensed sentence is anchored to the context of utterance, much like the reference of pronouns like *I* and *you*, or spatial and temporal adverbs like *here* and *now*. For example, to know when an event described by an utterance of a sentence like *John slept* or *John sleeps* takes place, we must first know the moment the sentence was spoken. After we specify this indexical nature of tense more clearly, we will compare two competing formal frameworks that account for it: the traditional operator-based approach pioneered by the logician Arthur Prior, and the more recent referential (or pronominal) approach.

An INDEXICAL (also known as a DEICTIC element) is "a word whose referent is dependent on the context of use, which provides

a rule which determines the referent in terms of certain aspects of the context" (Kaplan, 1977, 490). Examples include *I*, *my*, *you*, *here*, *now*, *tomorrow*, *yesterday*, *actual*, and *present*, and demonstratives accompanied by a gesture like *this* and *that*.

Formal foundations for the analysis of indexicals were laid out by Kaplan (1977), who showed how some of their logical properties could be captured by adding context as a parameter to the valuation function. In a Kaplanian system, rather than $[\![\phi]\!]^{M,g}$, the truth of a formula ϕ in Kaplan's logic is determined by a model M, an assignment function g, and a CONTEXT OF UTTERANCE c:[1]

$$[\![\phi]\!]^{M,g,c} = \ldots$$

The context of utterance determines who is speaking, to whom, when, and where.

$$c = \langle \mathsf{sp}, \mathsf{ad}, \mathsf{time}, \mathsf{loc} \rangle$$

An INDEXICAL CONSTANT in such a system is one whose value is determined by this context parameter c.[2] For example:

(1) a. $[\![\mathsf{i}]\!]^{M,g,c} = \mathsf{sp}(c)$
b. $[\![\mathsf{u}]\!]^{M,g,c} = \mathsf{ad}(c)$
c. $[\![\mathsf{now}]\!]^{M,g,c} = \mathsf{time}(c)$
d. $[\![\mathsf{here}]\!]^{M,g,c} = \mathsf{loc}(c)$

English pure indexicals can then be mapped to these special indexical constants like so:

(2) a. *I* $\rightsquigarrow$ i
b. *you* $\rightsquigarrow$ u

[1] The assignment function g, as before, determines the values of any free variables in ϕ. So expressions that are translated as variables get their meaning from the assignment function g, rather than the context of utterance c.

[2] Note: Although the reference of these terms is not fixed (since it varies by context), the logical expressions i, u, now, and here are classified as *constants* rather than *variables*, since they don't get their meaning from an assignment function.

c. *now* ⇝ now
d. *here* ⇝ here

Hence these pure indexical terms get their denotation from the context of utterance, rather than an assignment function.

In Kaplan's system, the truth or falsity of a formula depends among other things on the context of utterance. Since a proposition is something that can be either true or false, a sentence like *The queen is ill* does not express a proposition until the context of utterance fills in a value for the time of evaluation. If an utterance of *The queen is ill* takes place at 2pm on Thursday, August 10th, 2024, then the proposition expressed is a claim about that time. That claim will retain its truth value for eternity, even if she is sick on the 10th and recovers on the 11th. An utterance of the same sentence on the next day may express a different proposition.

Kaplan's distinction between 'content' and 'character' is helpful in sorting out what is similar and what is different across uses of the same sentence in different contexts.

- CONTENT is the proposition expressed by an utterance, with the referents of all of the indexicals resolved.

- CHARACTER is the aspect of meaning that two utterances of the same sentence share across different contexts of utterance.

An utterance of *The queen is ill* at 2pm on Thursday, August 10th, 2024, has as its *content* the proposition that the queen is ill on that day. But the *character* of the sentence *The queen is ill* is a function from contexts to propositions: given a context, this function returns the proposition that the queen is ill at the time of that context. The character of a sentence remains constant across all of its uses across different contexts.

Exercise 1. One might reasonably ask: Do the following two sentences have the same meaning or different meaning?

(3) a. (May 11, 2010, uttered by Elizabeth Coppock:)
I am turning 30 today.
b. (May 12, 2010, uttered by Elizabeth Coppock:)
I am turning 30 today.

Reasonable people might disagree. Similarly for the following pair:

(4) a. (May 11, 2010, uttered by Elizabeth Coppock:)
I am turning 30 today.
b. (May 12, 2010, uttered by Elizabeth Coppock:)
I turned 30 yesterday.

Again, reasonable people might disagree. Resolve this dispute with the help of Kaplan's concepts of content and character.

There is an alternative view one could consider. In fact, this alternative was incorporated into Richard Montague's original PTQ fragment (Montague, 1974b). One might say, in agreement with Arthur Prior, that the meaning of a sentence is a function from times to truth values. So then a sentence can change its truth value over time. For example, of the following sentence it might be said that it is true at one time, and then false later, after the queen has recovered:

(5) The queen is ill.

The stance according to which propositions can change their truth value over time is known as TEMPORALISM, and it is an idea that Prior formalized with his TENSE LOGIC. Priorian tense logic includes operators like 'Future' and 'Past' that can shift the time of evaluation. So if ϕ represents the meaning of (5), then $\mathsf{Future}(\phi)$

is true at time t if there is a time t' after t such that ϕ is true at t.

There are several reasons to reject the view of tense that is encoded in Prior's tense logic. Prior's analysis does not fit well with the actual details of the tense/aspect system in English. The closest analogue to Prior's 'Past' operator in English would be something like *It was the case that*, but this construction does not succeed in shifting the temporal interpretation of the sentences it embeds. Consider the following example (Kamp & Reyle, 1993, p. 496).

(6) It was the case that Mary is ill.

This sentence is odd, and is not synonymous with *Mary was ill*. The present tense in *Mary is ill* remains anchored to the time of utterance, rather than being shifted backwards in time by *It was the case that*.

A different example reinforcing the same point is the following.

(7) Fred told me that Mary is pregnant.

This sentence is not odd, but it can only be used to describe a scenario in which Fred made a claim about Mary being pregnant at the time of utterance. It cannot be used to describe a telling event that occurred several years prior to the time of utterance, unlike (8), which can:

(8) Fred told me that Mary was pregnant.

Both (6) and (7) show a tendency for the present tense to be anchored to the time of utterance.

The interpretation of future tense[3] is similarly impervious to embedding:

(9) It was predicted that the Messiah will come.

[3] We use 'future tense' descriptively here; as discussed in Section 11.2.4, *will* is more accurately analyzed as a modal rather than a genuine tense.

This sentence cannot be used to describe a time t in the past such that at time t, there was a prediction that the Messiah would come at some time t' in the future of t. (A scenario like that could be described by a variation on (9) with *would* in place of *will* — *would* being the past-tense form of the same modal.) Rather, the future tense remains anchored to the time of utterance, so that the sentence characterizes a prediction made in the past about a time following the 'now' of the utterance.

11.1.1.2 The anaphoric nature of tense

In addition to being deictic, tenses also appear to bear certain similarities to pronouns, being anaphoric to times that are salient in the discourse. For instance, the second sentence of (10) describes an event that directly follows the event described in the first sentence (Kamp & Reyle, 1993, ex. (5.21), p. 495).

(10) Bill left the house at a quarter past five. He took a taxi to the station and caught the first train to Bognor.

In Prior's tense logic, there is an operator 'Past' whose semantics is defined such that 'Past ϕ' is true at time t if *there is some time* t' prior to t such that ϕ is true at t'. This amounts to an *existential* theory of the past tense. Example (10) shows that the past tense contributes more than an existential claim about a time prior to the time of utterance; it contributes a claim about a particular contextually salient time.

Another piece of evidence suggesting that the past tense can behave like an anaphor is the following famous example due to Partee (1973). Consider a scenario in which you've just baked some cookies, and are on the way over to your friend's house. You realize mid-journey that you left the oven on. Then you say:

(11) Oh no! I didn't turn off the stove!

An existential theory of the past tense like Prior's does not make

correct predictions about this case. We could consider two possible scopes for negation relative to the past tense:

- Negation scopes over existential past tense (NOT > PAST): It is not the case that there is a time in the past when I turned off the stove.
- Existential past tense scopes over negation (PAST > NOT): There is a time in the past when I didn't turn off the stove.

Neither one of these is right. The first one is too strong – surely there is *some* time in the past when you turned off the stove. The second one is too weak – of course there is a time in the past when you didn't turn off the stove! For example, consider the moment you put the cookies in the oven; you didn't turn off the stove then. It seems that (11) is saying something *about a particular time*.

Partee (1973) notes a number of additional structural parallels between tenses and pronouns, in support of the so-called REFERENTIAL THEORY OF TENSE. On this view, the past tense in a sentence like (11) is similar to a free pronoun, anaphorically referring back to a time that has previously been introduced into the discourse.

Observations like Partee's suggest that tenses should be interpreted as variables over times, just as pronouns are interpreted as variables over individuals. And just as with pronouns, these variables can in principle be either free or bound. For example, as discussed by Heim (1994), Abusch (1988) treats the following case using existential quantification over the variable associated with past tense:

(12) John was in Paris at some time.

Klein (1994) mentions the following example, illustrating something more like (restricted) universal quantification over the variable associated with the past tense:

(13) Whenever I visited Carla, she was lying in bed.

With a treatment of tenses as variables that can potentially be bound, we can capture these effects.

11.1.1.3 Viewpoint aspect

Another desideratum for a theory of tense that a Prior-style theory fails to meet has to do with the interaction between tense and aspect, viewpoint aspect in particular (also known as 'grammatical aspect'). Viewpoint aspect can be thought of as locating events with respect to a point of view, and it interacts with tense to situate an eventuality in time.

English has two morphological forms that have been thought to express viewpoint aspect: the progressive (as in *I am eating*) and the perfect (as in *I have eaten*).

The perfect. The perfect can be found in multiple tenses, including present and past:

(14) a. I have seen it. (present perfect)
b. I had seen it. (past perfect)

In English, the perfect is realized as a combination of the verb *have* (in any form) and a verb phrase headed by an *-en* form (e.g. *seen*), known as the 'perfect participle'. (In many cases, the perfect participle is identical in form to the past participle, as in *I have mailed the letter*.) Compare (14a) and (14b) to the corresponding simple past sentence:

(15) I saw it. (simple past)

All three of these sentences describe an event of seeing something that precedes the time of utterance, but they all differ in meaning.

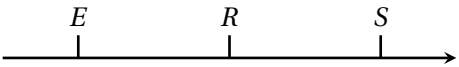
Reichenbach (1947) argued that in order to give a good theory of the English perfect, it is necessary to consider not only the time of utterance and the time at which the event occurred, but also

a more abstract time that he referred to as REFERENCE TIME (also called TOPIC TIME).

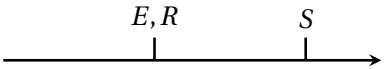
- SPEECH TIME (S): the time the sentence is uttered
- EVENT TIME (E): the time the event takes place
- REFERENCE TIME (R): the time under discussion

The concept of 'reference time' owes its currency in linguistics to Reichenbach (1947), though the idea itself is considerably older (see Klein, 1994, for a partial history). One way of characterizing it is that it is the time that the sentence is 'about' (hence the alternative term 'topic time').

According to Reichenbach, the difference between a past perfect sentence like (14b) and a simple past sentence like (15) is that in the former case, with the perfect, the sentence is about a time prior to speech time before which the seeing took place (so $E < R < S$):



In the case of the simple past, as in (15), the sentence is about the time at which the seeing took place (so $E = R < S$), a time prior to speech time.

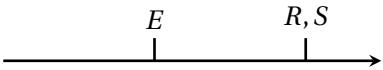


Aside from its intuitive appeal, support for this analysis comes from the fact that temporal adverbs like *at 5pm* track the hypothesized reference time:

(16) a. At 5pm, I mailed the letter. (5pm = mailing time)
b. At 5pm, I had mailed the letter. (mailing time < 5pm)

Assuming the modifier is identified with the reference time, we correctly predict that 5pm is the time of mailing the letter for the sentence in simple past, but a time before the speech time and after the time of mailing the letter in the past perfect.

In the past tense, then, the perfect splits apart the reference time and the event time, and locates the event time prior to the reference time. It has the same effect in the present tense as well. Consider (14a), *I have seen it*. Assuming that in the present tense, the reference time is the time of utterance and that the perfect locates the event time prior to reference time yields the following diagram for the present perfect:



This analysis correctly locates the time of seeing prior to the time of utterance. Unlike in the simple past and the past perfect, however, the reference time is identified with the time of utterance. If so, then a temporal modifier like *now* should be combinable with the present perfect, since temporal modifiers pick out the reference time. This prediction is borne out:

(17) Now I have mailed the letter.

Compare:

(18) a. ??Now I had mailed the letter.
b. ??Now I mailed the letter.

We can explain the degraded nature of (18a) and (18b) as follows: Both are past tense sentences, requiring the reference time to be prior to the time of utterance. Use of *now* as a sentence-initial temporal modifier imposes the requirement that the reference time be identified with the time of utterance. These two requirements conflict with each other.

This analysis of the perfect makes good predictions about the future tense as well. Assuming that the future tense locates the

reference time after speech time, and that the perfect locates the event time prior to the reference time, a sentence like:

(19) At 5pm, I will have mailed the letter.

is predicted to imply that 5pm is in the future, and that mailing the letter will have occurred prior to that (perhaps before, or perhaps after speech time). This accords with intuition. In contrast:

(20) At 5pm, I will mail the letter.

implies that the letter-mailing is in the future.

Exercise 2. With the tools just developed, explain the following contrast.

(21) a. Now I have mailed the letter.
b. #Now I mailed the letter.

The picture we arrive at, then, is that the contribution of tense is to relate the reference time with the speech time, and the contribution of the perfect is to relate the event time to the reference time. Past tense locates the reference time prior to speech time; present tense sets them equal; and future locates reference time after speech time. The perfect places the event prior to reference time; otherwise the event takes place at reference time.

This view is somewhat of an oversimplification; there are a number of different uses for the English perfect, including:

(22) a. Ed has put the cake in the oven. RESULTATIVE
b. Ed has visited Korea many times. EXISTENTIAL
c. Ed has lived in Korea for 3 years. UNIVERSAL

We set these uses aside, but see Bhatt & Pancheva (2005) for a good overview of the issues involved, and arguments for a so-called 'extended now' theory of the perfect (McCoard, 1979), on which the

perfect locates the event within an interval that stretches back from the speech time into the past.

Perfective and imperfective aspect. One broad distinction in the realm of viewpoint aspect is between PERFECTIVE and IMPERFECTIVE aspect. Perfective aspect "looks at the situation from outside, without necessarily distinguishing any of the internal structure of the situation, whereas the imperfective aspect looks at the situation from inside, and as such is crucially concerned with the internal structure of the situation" (Comrie, 1976, 4). Imperfective aspect coresponds to a category of grammatical forms associated with a lack of completion, a continuous state, an iterated sequence of events, or a habitual pattern. This distinction is grammatically marked in a number of languages, including Romance and Slavic languages, illustrated in the following examples from Spanish and Russian, respectively.

(23) a. Juan leyó el libro.
Juan read.PAST.PERFECTIVE.3SG the book
'John read the book.'

b. Juan leía el libro.
Juan read.PAST.IMPERFECTIVE.3SG the book
'John was reading the book.' / 'John used to read the book.'

(24) a. Ivan sjel sup
Ivan eat.PAST.PERFECTIVE.MASC soup
'Ivan ate up (all) the soup.'

b. Ivan jel sup
Ivan eat.PAST.IMPERFECTIVE.MASC soup
'Ivan was eating the/some soup.' / 'Ivan used to eat the/some soup.'

As the translations of the imperfective examples (23b) and (24b) show, the past progressive (*was V-ing*) only captures one of the meanings that the past imperfective can have in French and Rus-

sian. Another kind of interpretation that imperfective aspect has in those languages is a habitual one, expressed more effectively by *used to* than by the progressive in English.

The fact that the English progressive has a more restricted range of interpretations than imperfectives in other languages is one reason to agree with Comrie (1976, 25) that the English progressive is a special subtype of imperfective aspect. Another is that English progressive forms are more restricted in their distribution than imperfective forms in languages like French and Russian. The English progressive is sensitive to the distinction between stative and dynamic predicates, a distinction that falls under the heading of AKTIONSART.[4] States are static — they hold at instants and require no influx of energy — while DYNAMIC predicates describe changes or actions.[5] The progressive is not regularly used with stative predicates:

(25) a. Ida is running along the beach. [dynamic]
b. ??Ida is having a friend. [stative]

And the simple present gives a habitual interpretation only with non-stative predicates:

(26) a. Ida runs along the beach. [dynamic: habitual]
b. Ida has a friend. [stative: non-habitual]

What is the semantics of the progressive? Consider the following contrast:

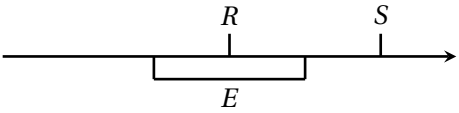
(27) a. At 5pm, I mailed the letter. (simple past)

[4] Vendler (1957) proposed a four-way classification of eventualities into states, activities, accomplishments, and achievements, based on the dimensions of stativity, telicity, and durativity; a fifth type, semelfactives, was later added (Smith, 1997). Other terms for *aktionsart* include *lexical aspect*, *situation aspect*, and *aspectual class*.

[5] The term 'event' is sometimes used broadly to cover both states and events, and is sometimes used in contrast to states, with the more neutral term EVENTUALITY covering both.

b. At 5pm, I was mailing the letter. (past progressive)

In the first case, there was a letter-mailing event that took place at 5pm. In the second case, there was a letter-mailing event in progress at 5pm which may not have completed by that time. Bennett & Partee (1972) propose to model this contrast using inclusion among time intervals. In the past progressive example (27b), the time interval during which the letter-mailing event took place includes the reference time identified with *5pm*. This gives us the following picture for past progressive:



Perfective aspect is often analyzed as the reverse of what Bennett & Partee (1972) propose for the progressive: the reference time contains the event time. This notion of 'containedness' entails a view of times on which they are actually stretches of time, or time INTERVALS. A time t contains another time t' if every moment included within t' is also included in t. Using $\subseteq$ to represent this containment relationship, we can represent the contribution of aspectual morphology as follows:

- perfective aspect: $E \subseteq R$
- progressive aspect: $R \subseteq E$

This view captures the contrast in (27b); the non-progressive sentence implies that a letter-mailing event occurred at 5pm, while the progressive sentence does not.

There are a number of challenges for this view, which we will not attempt to address. Observe, for example, that progressive and non-progressive past tense sentences differ in their entailment patterns. A past tense sentence implies that the event was completed, while a progressive sentence does not.

	PERFECTIVE	IMPERFECTIVE
PERFECT	*I have danced*	*I have been dancing*
NON-PERFECT	*I danced*	*I was dancing*

Table 11.1: Aspectual distinctions in English

(28) a. I walked to the park ⇒ I got to the park.
b. I was walking to the park ⇏ I got to the park.

The $R \subseteq E$ analysis of progressive does not require completion of the event in the past, so it correctly predicts that *I was walking to the park* does not entail *I got to the park*, but it still implies that there is an event of me walking to the park; that at some point the event will become complete. So, as Parsons (1990) and Dowty (1979) point out, this analysis predicts that the following argument should be valid:

(29) Mary was building a house. Therefore, Mary will have built a house.

See Bhatt & Pancheva (2005), lecture 4 for a more thorough discussion of what is at stake in the analysis of the progressive, what the various options are, and their advantages and disadvantages. For simplicity, we will stick with the Bennett and Partee analysis, acknowledging that there is much more to the story.

Terminological note: Due to an unfortunate accident in the development of standard terminology in this domain, the term 'perfective' is *not* an adjectival form of 'perfect'. In fact, these categories can cross-classify; see Table 11.1.

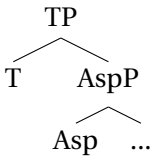
11.2 English tense and aspect

We now sketch a theory of tense and aspect in English, moving from the innermost layer of the syntactic structure out to the tense layer. As a first step, let us establish our assumptions about the syntactic structure of tense and aspect.

11.2.1 The syntax of tense and aspect in English

Kratzer (1998) proposes that the syntax of verb phrases is layered so that an aspectual phrase AspP dominates the VP, and a tense phrase TP in turn dominates the aspectual phrase:

(30)

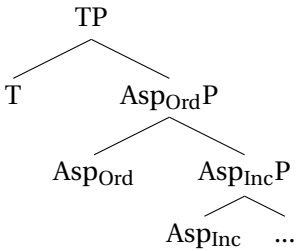


This single-layer structure cannot account for the fact that the perfect and the progressive can be stacked:

(31) a. I have been reading it.
b. I had been reading it.
c. I will have been reading it.

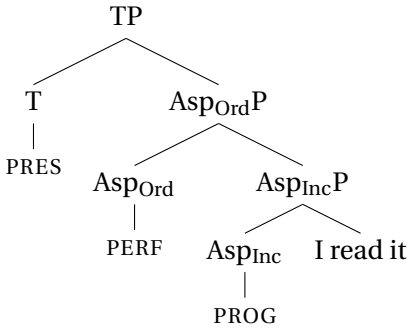
We therefore assume two aspect projections between tense and VP (McCoard, 1979; Dowty, 1979; Iatridou et al., 2001; Bhatt & Pancheva, 2005; Rullmann & Matthewson, 2017). Following Rullmann & Matthewson (2017), we call the outer layer *ordering aspect* and the inner layer *inclusion aspect*:

(32)



Thus (31a) has the structure:

(33)

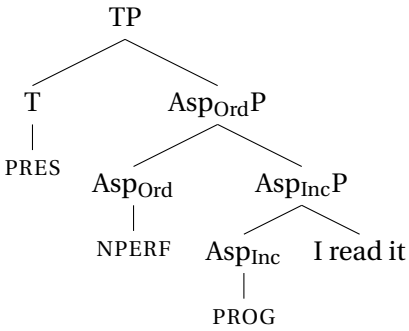


The ordering aspect projection is always present. In sentences with the overt perfect morpheme *have*, it is headed by the overt morpheme PERF. In all other sentences — simple past, simple present, progressive — it is headed by the covert morpheme NPERF. Thus a present progressive sentence like:

(34) I am reading it.

has the structure:

(35)



Our task now is to provide translations into L_T for elements that can head tense/aspect projections that allow for them to combine compositionally, and make correct predictions about the interpretations that arise when they do.

11.2.2 Semantics of vPs

In the fragment of English that we treat in this chapter, verbal predicates take time arguments along with arguments for the participants in the eventuality they describe. For example:

(36) *dance* $\rightsquigarrow \lambda x.\lambda t.\mathsf{dance}_t(x)$

(37) *read* $\rightsquigarrow \lambda y\lambda x.\lambda t.\mathsf{read}_t(x,y)$

This expression denotes a function from individuals to functions from times to truth values. We assume that the individual participant arguments combine with the verb prior to any tense/aspect information at the vP level. Assuming that *Ann* $\rightsquigarrow \mathsf{a}$, *Ann read* it_i will be interpreted as:

(38) $\lambda t.\mathsf{read}_t(\mathsf{a}, x_i)$

11.2.3 Inclusion aspect

Let us assume, following Bennett & Partee (1972), that the English progressive form signals that the event time includes the refer-

ence time. The vP that an Asp head combines with provides a predicate of times. Times that fall under this predicate can be thought of as 'event times' because they are times at which an event of the kind described by the non-finite part of the sentence takes place. The PROG head takes this 'event time predicate', as we might call it, and produces a new predicate of times, ones which can play the role of reference time. Specifically:

(39) PROG $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\exists t'.t \subseteq t' \wedge P(t')$
'Takes a predicate of times P, and returns a predicate of times that is true of a time t if t is contained in a time t' at which P is true.'

We treat perfective aspect in terms of inclusion as well, but in the other direction:

(40) PFV $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\exists t'.t' \subseteq t \wedge P(t')$
'Takes a predicate of times P, and returns a new predicate of times that is true of a time t if t contains a time t' at which P is true.'

Using the assumptions made so far (and assuming that the English word *I* is translated as the indexical constant i, and adopting a Fregean analysis of the definite article), we obtain the following translation into L_T at the Asp_{Inc}P node for *I am mailing the letter*:

(41) [$_{\text{Asp}_{\text{Inc}}\text{P}}$ PROG [$_{\text{vP}}$ I mail the letter]]
$\lambda t.\exists t'[t \subseteq t' \wedge \text{mail}_{t'}(\text{i}, \iota x.\text{letter}(x))]$

This is a predicate of times that holds of a time t if there is a time t' containing t at which the speaker mails the letter.

We assume that non-progressive sentences are interpreted as perfective. Hence the Asp_{Inc}P node of a non-progressive clause like *I mailed the letter* would be translated as follows:

(42) [$_{\text{Asp}_{\text{Inc}}\text{P}}$ PFV [$_{\text{vP}}$ I mail the letter]]

$$\lambda t.\exists t'[t' \subseteq t \wedge \mathsf{mail}_{t'}(\mathsf{i}, \iota x.\mathsf{letter}(x))]$$

This is a predicate of times that holds of a time t if there is a time t' that t contains at which the speaker mails the letter.

11.2.4 Ordering aspect

As introduced above, ordering aspect encodes the temporal ordering relation between the reference time and the time at which the inclusion-aspect predicate holds. English has two ordering aspect morphemes. The overt one is the perfect morpheme *have*:

(43) PERF $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\exists t'.t' < t \wedge P(t')$
'Takes a predicate of times P, and returns a predicate of times that is true of a time t if t follows t' at which P is true.'

The covert ordering aspect morpheme NPERF is the default; it heads Asp_{Ord}P in all sentences that do not contain an overt *have*. NPERF is semantically transparent — it passes Asp_{Inc}P through without modification, letting the reference time directly parameterize the inclusion aspect:

(44) NPERF $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.P(t)$
'The identity function on predicates of times: the event time is simultaneous with the reference time.'

The non-perfect thus places no additional ordering constraint on the event time beyond what Asp_{Inc}P itself specifies.

11.2.5 Tense

Let us now outline a simple theory of tense.

Past tense. Following Partee (1973) and others, we adopt a PRONOMINAL THEORY OF TENSE (Bochnak, 2016). The Tense head T bears

an index n, just like a pronoun, and its denotation is the corresponding time variable t_n, which serves as the REFERENCE TIME of the clause:

(45) $T_n \rightsquigarrow t_n$
(temporal pronoun, type i)

The value of t_n comes from the assignment function g.

We introduce a predicate of times, following the same pattern as gender features on pronouns (Chapter 8): past, of type $\langle i, t\rangle$. This predicate is true of times that precede the speech time: $\mathsf{past}(t) := t < \mathsf{now}$.

Now the question becomes how this predicate comes to constrain a salient time in the context and construe it as the reference time — the time corresponding to the lambda-bound variable on the ordering aspect node. One possibility is to let PAST be a function of a time t (provided in the syntax by a silent temporal pronoun) and a predicate of times T (provided in the syntax by the highest aspect projection), asserting a conjunction: the input time precedes the time of utterance, and the predicate holds of that time.

(46) $\text{PAST}_{\text{ent}} \rightsquigarrow \lambda T \lambda t . [T(t) \wedge \mathsf{past}(t)]$
(past tense, entailment hypothesis, type $\langle\langle i, t\rangle, \langle i, t\rangle\rangle$)

Notice that under the foregoing analysis of the past tense morpheme, it is an ordinary entailment rather than a presupposition that the salient time is in the past. Call (46) the PAST-AS-ENTAILMENT ANALYSIS. But one might imagine this content to be part of the presupposed content instead. Formally, the past-as-presupposition analysis can be represented as follows. Using the superscript convention from Chapter 8, $t^{\mathsf{past}} = t_{|\mathsf{past}(t)} = t_{|t<\mathsf{now}}$: the time t presupposed to be in the past. PAST is then a partial identity function on times that maps t to t^{past}, imposing the presupposition as a condition on its input, in parallel with gender features on pronouns (Heim, 1994):

(47) PAST $\rightsquigarrow \lambda t\,.\,t^{\mathsf{past}}$
(past tense, presupposition hypothesis, type $\langle i, i\rangle$)

How might one distinguish between these two hypotheses? One standard test for whether a piece of content is presupposed rather than asserted is to check whether it survives being negated: presupposed content typically PROJECTS past negation, while asserted content does not. Applying this test to $t < \mathsf{now}$ requires an assumption about how negation and tense scope relative to each other. In our grammar, negation combines with the untensed predicate contributed by the verb phrase and the aspect projections (Section 6.5); the past-tense content is only introduced afterward, when this predicate combines with the tense-marked reference time. So negation is structurally trapped below the level at which pastness is introduced: tense always outscopes negation, never the other way around.

If tense is guaranteed to outscope negation, then the use of a negated sentence like *I didn't turn off the stove* could not provide evidence for or against the presuppositional status of the pastness conjunct. Fortunately, there is a more general argument, based on the semantics of yes/no questions rather than declarative negation. Answering *no* to a yes/no question targets the negation of the entire proposition expressed by the question, so that *yes* and *no* stand in contradictory opposition: exactly one is true, the other false. Unlike a lexical item such as *not* embedded inside a declarative, a bare particle like *no* has no comparable scope ambiguity — it simply negates the whole proposition.

(48) Did you turn off the stove? — No.

Suppose (46), the entailment hypothesis, is correct. Now imagine that you did turn off the stove five minutes before speaking, but the time t that is contextually salient — say, the moment you are about to leave the house — happens to lie in the future. Then $T(t)$ is false (nothing relevant happens at t itself) and $\mathsf{past}(t)$

is also false (since t is not in the past), so the negated proposition $\neg[T(t) \wedge \mathsf{past}(t)]$ comes out true purely on the strength of the second conjunct, regardless of whether the stove was actually turned off. Under (46), *no* would incorrectly count as a true, felicitous answer.

This argument relied on the assumption, established above, that tense outscopes negation, which is what makes the question-answer test necessary in the first place: it lets us negate the whole proposition without needing to embed *not* inside a declarative. But suppose instead that negation can outscope tense after all. Then the same problem simply resurfaces directly in the declarative sentence: *I didn't turn off the stove* would have exactly the same truth conditions as our *no* answer above, $\neg[T(t) \wedge \mathsf{past}(t)]$, and so would be wrongly predicted true in exactly the scenario that made *no* wrongly felicitous. Either way — whether tense outscopes negation or negation outscopes tense — (46) makes the wrong prediction. (47), the presupposition hypothesis, avoids the problem under both assumptions: the question, or the declarative, simply lacks a defined answer when t is not in the past, rather than incorrectly licensing *no*. We therefore adopt (47) going forward.

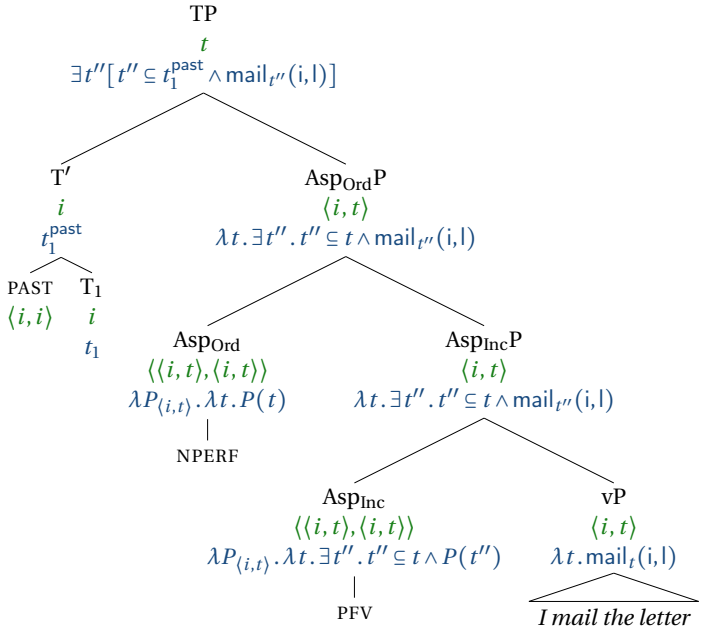
The tense feature and the temporal pronoun together form the T′ node, which denotes t_n^{past}: the reference time t_n presupposed to precede the speech time. The Asp_{Ord}P node denotes a predicate of times (type $\langle i, t\rangle$); it is applied to T′ to yield the TP:

$$\text{TP} = \text{Asp}_{\text{Ord}}\text{P}(t_n^{\mathsf{past}})$$

This is a proposition (type t) with free variable t_n, carrying the presupposition $\mathsf{past}(t_n)$, i.e., that $t_n < \mathsf{now}$.

Thus for an example like *I mailed the letter*, we have the derivation in (49).

(49)



At the top node, t_1 is a free variable over times, presupposed to precede the time of utterance. The discourse context provides an assignment function giving a value to t_1; as long as that value precedes the time of utterance, the sentence has a defined truth value. In other words, the expression has a defined value relative to assignment function g and context of utterance c as long as $g(t_1)$ precedes the time of utterance $t(c)$.

Exercise 3. Write out how you would read the expression at the top of (49) aloud.

Exercise 4. Explain how the treatment of tense and aspect that we have built up so far captures the 'completion inference' of the past perfective, i.e., the fact that *I mailed the letter* implies that there is

a letter-mailing event carried out by the speaker that has reached completion.

Exercise 5. Compute a tree for *I was mailing the letter.* Does the formula you derive carry a completion inference? Explain why or why not.

Present tense. Following Rullmann & Matthewson (2017), we analyze the present tense in parallel to the past. We first introduce a predicate of times pres (type $\langle i, t\rangle$), true of times that coincide with the speech time: $\mathsf{pres}(t) := t = \mathsf{now}$. By the superscript convention, $t^{\mathsf{pres}} = t_{|\mathsf{pres}(t)}$: the time t presupposed to equal the speech time. The T head then bears a PRES feature (type $\langle i, i\rangle$), a partial identity function that maps t to t^{pres}:

(50) PRES $\rightsquigarrow \lambda t\,.\,t^{\mathsf{pres}}$
(tense feature, type $\langle i, i\rangle$)

$\mathrm{T}' = \textsc{pres}(T_n) = t_n^{\mathsf{pres}}$. Then *I am mailing the letter* has the derivation:

(51) [$_{\mathrm{TP}}$ PRES T_n [$_{\mathrm{Asp_{Ord}P}}$ NPERF [$_{\mathrm{Asp_{Inc}P}}$ PROG [$_{\mathrm{vP}}$ I mail the letter]]]]
$\exists t'[t_n^{\mathsf{pres}} \leq t' \wedge \exists t''[t' \subseteq t'' \wedge \mathsf{mail}_{t''}(\mathsf{i},\mathsf{l})]]$

The presupposition on t_n^{pres} requires the reference time to equal the speech time ($t_n = \mathsf{now}$); together with the ordering condition $t_n^{\mathsf{pres}} \leq t'$, this forces the event window t' to be at or after now. This gives the familiar progressive reading: the mailing event is in progress at the time of utterance. The event need not be complete by the time of utterance; it is merely in progress.

Exercise 6. Give a derivation tree for *I am mailing the letter*.

Now, in English, the simple present tense cannot be used with dynamic predicates to describe an event that is currently happening:

(52) Q: What are you doing?
A: #I mail the letter.

With dynamic predicates, the simple present gives rise to an iterated or habitual interpretation (e.g. *Whenever I finish writing to my mother, I mail the letter right away*). Statives are the only kind of predicates that give rise to a non-habitual interpretation in English (e.g. *I love Paris*, *I live in Paris*, etc.).

In this respect, English contrasts with many other languages, even its close relative, German. Under the ordering aspect analysis adopted here, NPERF is the identity function, so the simple present with PFV yields $\text{PFV}(\upsilon P)(t_n) = \exists t''[t'' \subseteq t_n \wedge \upsilon P(t'')]$. Assuming the time of utterance is an instant, PFV requires the event to be contained within t_n = now; since no extended event can be contained in an instant, the sentence is infelicitous. Stative predicates, which hold at points, are not subject to this constraint, which is why *I love Paris* is fine under PRES. Future-oriented simple-present readings (*The train leaves at 5pm*) are derived with WOLL, the prospective aspect head introduced below, as the ordering aspect head rather than NPERF.

Exercise 7. Give a derivation tree for *I mail the letter* on which it involves perfective aspect, and assume that the time of utterance is a mere instant of time. Does this analysis explain the awkwardness of (52)? If so, why? If not, what additional or alternative assumptions could you adopt in order to explain it?

There are quite a number of additional puzzles regarding the present tense that we are not dealing with here. For instance, the present tense behaves somewhat differently from the word *now* (Kamp, 1971). Compare:

(53) a. Someday Susan will marry a man she loves.
b. Someday Susan will marry a man she loves now.

These two sentences mean something different; the former describes a man she will love in the future; the latter describes a man she loves now. This contrast can be captured using Kratzer's (1998) notion of 'zero tense'. A 'zero tense' for Kratzer is an indexed time variable with no presuppositions (hence the name 'zero'), which must be bound by a local antecedent.[6] We will maintain the simple theory of the present tense in (50) for the time being, though.

Future... tense? Now for the future. It is natural to suppose that the English verb *will* is a tense that relates to a time located after utterance time. Using this idea, the lexical entry that is analogous to the ones we have given for past and present would be as follows:

(54) $\text{FUT} \rightsquigarrow \lambda P_{\langle i,t\rangle} \lambda t\,.\,[t > \text{now} \wedge P(t)]$

However, *will* is widely analyzed as having modal rather than purely temporal semantics (Chomsky, 1957; Partee, 1973, i.a.). According to this view, what appears to be a future tense is actually a combination of the present tense with this modal. This idea goes back at least to Jespersen (1914/1970) (see also Cable 2008 for recent discussion); evidence for it comes from the fact that *will* has a past

[6] Kratzer (1998) analogizes zero tenses to the phenomenon observed in sentences like *Only I did my homework*, where the first person possessive pronoun *my* seems to be interpretable without its first person feature: the sentence is ambiguous between 'I am the only person x such that x did x's homework' (where *my* lacks its first person feature) and 'I am the only person x such that x did my (the speaker's) homework' (where it retains it).

tense variant, *would*:

(55) (In 1981, Dave's marriage was very stable.)
However, he **would** later learn (in 1987) that his wife was cheating on him.

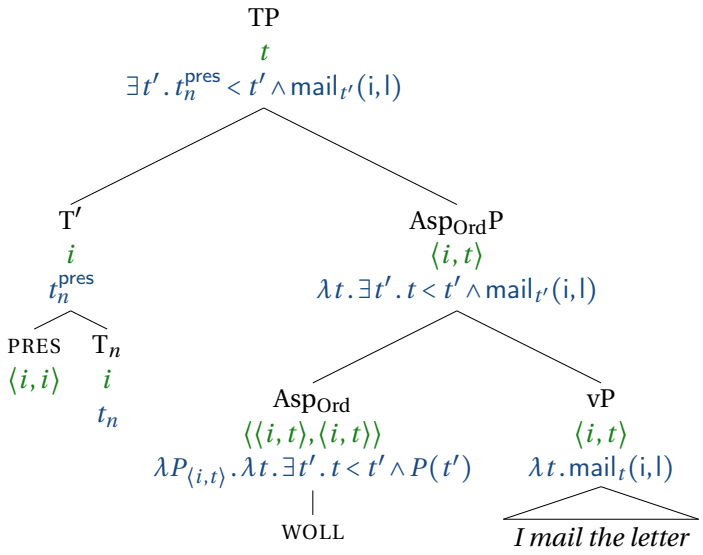
This sentence means that, spoken in 1981, the sentence "Dave **will** learn that his wife **is** cheating" is true. If there is a past version of *will*, then *will* must represent the combination of two elements, a tense element and something else.

Following Abusch (1997), let us suppose that *will* and *would* are present and past versions, respectively, of an underlying verb WOLL, defined as follows:

(56) $\text{WOLL} \rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.[\exists t'.t < t' \wedge P(t')]$

This entry has the form of a *prospective aspect* operator: it asserts that the predicate holds at some future time. WOLL can combine with both present and past morphology, so the 'future' is not a tense in English. Prospective aspect is also attested as an overt morpheme in languages such as Gitksan and St'át'imcets (Matthewson, 2006). Following Rullmann & Matthewson (2017), we implement this modal analysis by treating WOLL as an ordering aspect head (Asp_{Ord}), projecting an Asp_{Ord}P in the same position as NPERF and PERF, deferring a full treatment of modality and world-quantification to Chapter 12. The sentence *I will mail the letter* will have the representation in (57), ignoring the possibility that it may have perfective aspect.

(57)

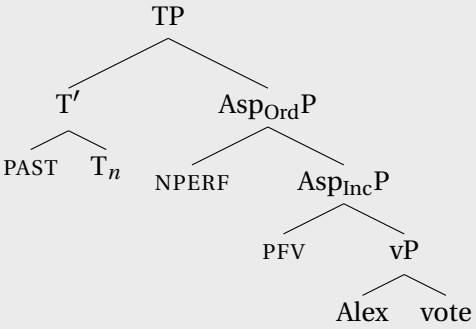


Exercise 8. Assume that *I will mail the letter* involves perfective aspect, and give a derivation of the truth conditions for that sentence that incorporates this assumption.

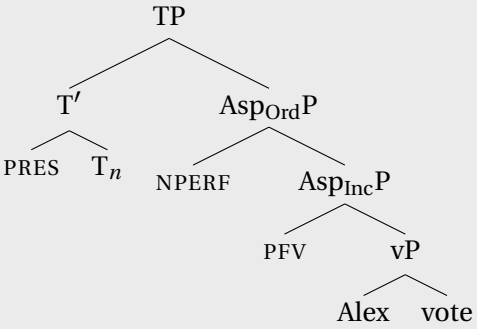
Exercise 9. English tense and aspect.

Compositionally derive meanings for the following sentences.

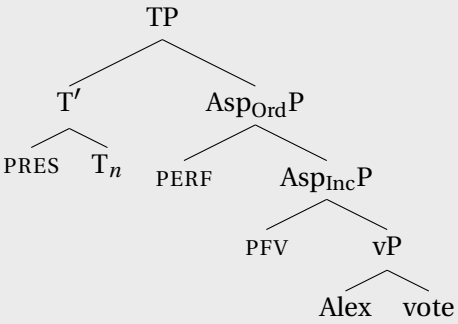
(i) Alex voted



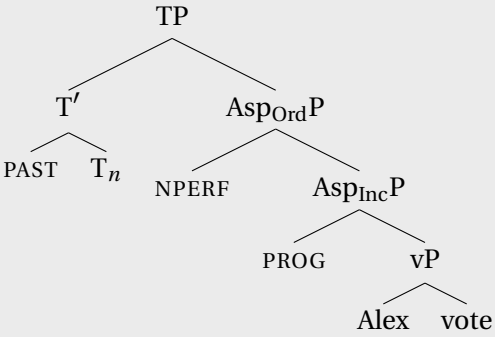
(ii) Alex votes



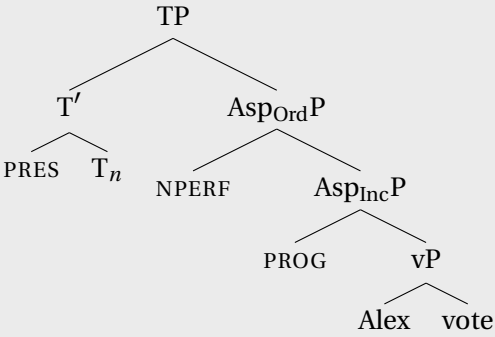
(iii) Alex has voted



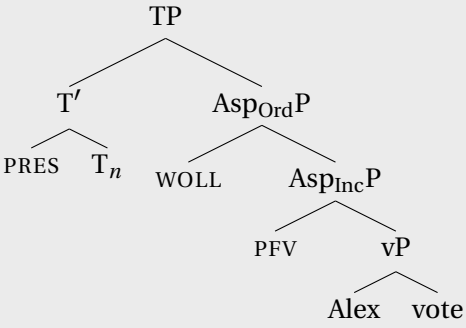
(iv) Alex was voting



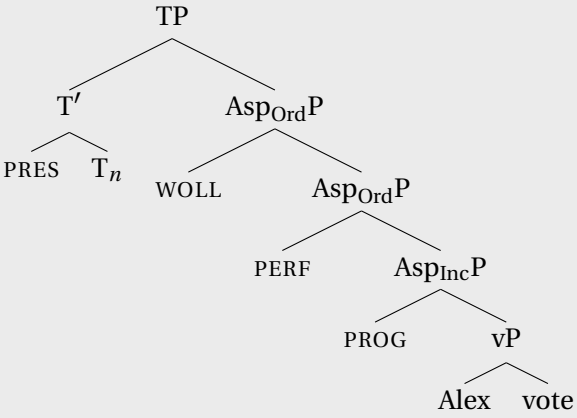
(v) Alex is voting



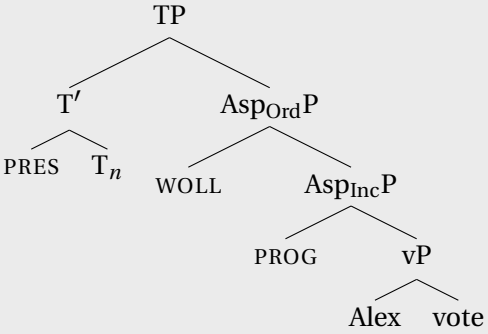
(vi) Alex will vote



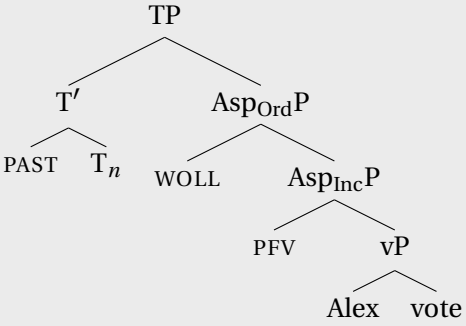
(vii) Alex will have been voting



(viii) Alex will be voting



(ix) Alex would vote



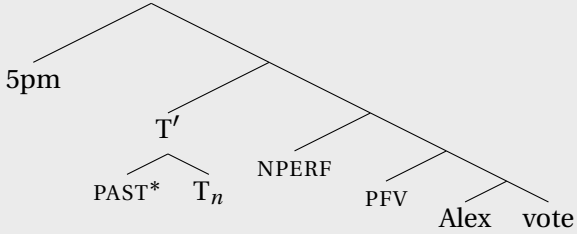
Exercise 10. Taking modifiers into account.

Give a new definition for the past tense, represented here as PAST*, that would allow the sentence to take a temporal modifier like *at 5pm*.

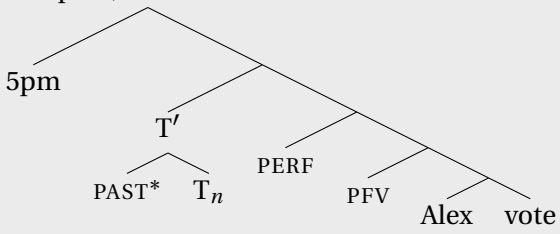
After providing the new definition, compositionally derive the following sentences with it.

- PAST* $\rightsquigarrow$?

(i) Alex voted at 5 p.m..



(ii) At 5 p.m., Alex had voted.



11.3 A formal theory of tense

11.3.1 A partial, context-sensitive logic with times

11.3.1.1 Syntax

We define a representation language L_T, a logic with time-denoting expressions with context-sensitivity and presuppositions. We use i as the type designator for times, so expressions that refer to times will be of type i. L_T adds variables of type i: the formal name for the variable with index n is $v_{n,i}$. We write t_n as an abbreviation for $v_{n,i}$ when the index carries semantic content, and t, t', t'' for arbitrary variables of type i. When multiple such symbols appear in the same formula, they are understood to be distinct from one another (distinct as variables, not necessarily as values).

L_T contains the symbols $<$ and $\subseteq$, standing for temporal precedence and temporal inclusion:

(58) Syntactic rule ($<$)
If α and β are expressions of type i, then $\alpha < \beta$ is a formula.

(59) Syntactic rule ($\subseteq$)
If α and β are expressions of type i, then $\alpha \subseteq \beta$ is a formula.

The expression $t \subseteq t'$ can be read, 't is contained in t''. Thus t' is the (potentially) larger interval, occupying a stretch of time that contains the stretch of time t occupies. (We'll get to the semantic definition in Section 11.3.1.2.)

L_T includes predicates expressing relations between individuals and times. For instance, the following is a formula of type t which says that x is ill at time t:

(60) $\mathsf{ill}(x)(t)$

(Of course, being ill is a gradient and multidimensional matter, as for example Sassoon (2013) discusses, but we gloss over that subtlety here.) As a notational convention, we will place the temporal

argument to a predicate in a subscript, like so:

(61) $\mathsf{ill}_t(x)$

L_T contains a number of special indexical constants, whose interpretation is relative to a given context of utterance. These include now, denoting the time of utterance, i, denoting the speaker of the context, and u, denoting the addressee of the context. now is an expression of type i, so the following formula is well-formed:

(62) $\mathsf{ill}_{\mathsf{now}}(\mathsf{q})$

This formula expresses that, at the time of utterance, there is a state of being ill held by the queen.

When we say "a state of being ill held by the queen", we mean a state that meets a certain description. As Klein (1994) reminds us, there may be many different particular states that meet the description at the relevant time, and it is important to draw a clear distinction between the state itself and a description thereof. The sentence does not require that any particular eventuality obtains at the time of utterance, only that there *is* one that matches the description. We will use:

(63) $\mathsf{ill}_t(\mathsf{q})$

to express the existential claim that there was a state of being ill held by the queen whose full temporal extent was t. We are not explicitly appealing to eventualities in this chapter, but if we were, we would write (63) as:

(64) $\exists e.\mathsf{ill}(e) \wedge \mathsf{holder}(e,\mathsf{q}) \wedge \tau(e) = t$

Here, $\tau(e)$ can be read as 'the temporal trace of e', that is, the time interval corresponding to e's spatial extent. Although we will not make our variables over eventualities explicit in this chapter, we may still think of the semantics of predicates like *ill* in terms of them.

Otherwise, the syntax of L_T is just the same as L_∂, with the same connectives and variable binders, including not only the familiar devices of lambda abstraction and quantification but also the iota operator.

11.3.1.2 Semantics

The notion of 'speech time' is an INDEXICAL one: It has to do with the so-called CONTEXT OF UTTERANCE, or CONTEXT OF USE. The 'context of utterance' is the here and now of the utterance: who is speaking, to whom, where, when, etc. To model tense we will use an extension of Kaplan's system as described in Section 11.1.1.1, where the semantic value of an expression is determined relative to a model M, an assignment function g, and a context of utterance c.

$$[\![\alpha]\!]^{M,g,c}$$

The 'utterance time' is the time determined by c, which we call $t(c)$.

The semantic value of an expression is defined relative to a TEMPORAL MODEL M, which specifies a set T of time intervals along with a set of individuals D. Let $\mathcal{T}$ be the set of types (e for individuals, t for truth values, i for times, $\langle e, t\rangle$ for functions from individuals to truth values, etc.). As usual, for each type $\tau \in \mathcal{T}$, the model determines a corresponding domain D_τ. We define the standard frame based on D and T as an indexed family of sets $(D_\tau)_{\tau\in\mathcal{T}}$, where:

- $D_e = D$
- $D_i = T$
- $D_t = \{\mathsf{T}, \mathsf{F}\}$
- for any types σ and τ, $D_{\langle\sigma,\tau\rangle}$ is the set of functions from D_σ to D_τ

As in L_∂ from Chapter 8, models are associated with an AUGMENTED FRAME along with the standard frame. As in that chapter, we distinguish here between CLASSICAL DOMAINS D_τ and FIXED-UP DOMAINS D^+_τ, for any type τ. The fixed-up domains include the undefined entities. For atomic types, the fixed-up domains are defined as the union of the classical domains with the undefined entity of the corresponding type:

- $D^+_t = (D_t \cup \{\#_t\})$.
- $D^+_e = (D_e \cup \{\#_e\})$.
- $D^+_i = (D_i \cup \{\#_i\})$.

As before, for complex types $D^+_{\langle\sigma,\tau\rangle}$, we define $D^+_{\langle\sigma,\tau\rangle}$ as the set of functions from D^+_σ to D^+_τ, and we define the undefined entity of any complex type $\langle\sigma,\tau\rangle$ as a function whose output is always $\#_\tau$, the undefined entity of type τ.

A temporal model M for L_T based on a set of individuals D and a set of times T is a tuple:

$$M = \langle (D_\tau)_\tau, (D_\tau)^+_\tau, I, <, \subseteq, C\rangle$$

where

- $(D_\tau)_\tau$ is standard frame based on D and T
- $(D_\tau)^+_\tau$ is an augmented frame based on D and T
- for every type $\tau \in \mathcal{T}$, I assigns to every non-logical constant of type τ an object from the domain D^+_τ
- $<$ is a precedence relation among elements of T, which is a linear order – transitive, irreflexive, and total (so for every pair of times t_1 and t_2, either $t_1 < t_2$ or $t_2 < t_1$)
- $\subseteq$ is a containment relation among elements of T

- C is a set of CONTEXTS OF UTTERANCE; each $c \in C$ is a tuple $c = \langle \mathsf{sp}(c), \mathsf{ad}(c), \mathsf{time}(c), \mathsf{loc}(c) \rangle$ where $\mathsf{sp}(c) \in D$ is the speaker, $\mathsf{ad}(c) \in D$ is the addressee, $\mathsf{time}(c) \in T$ is the time of utterance, and $\mathsf{loc}(c) \in D$ is the location.

The idea that time is linearly ordered may seem obvious, but there are alternatives. On BRANCHING TIME model, there can be multiple timelines that branch out from a given point, although there is a unique sequence of moments preceding a given time point (see for example Goranko & Rumberg 2023). On this view, it is possible to have a pair of times t_1 and t_2 that are not ordered via the precedence relation; in other words, neither $t_1 < t_2$ nor $t_2 < t_1$ holds. von Prince (2019) argues that a branching time model is ideal for capturing the relationship between counterfactuality and past tense marking.[7] For now, we will stick to the simple view that there is only one timeline.

Semantics for the connectives and variable binders inherited from L_∂ are just as in L_∂. The semantics for the precedence and inclusion symbols of L_T are defined in terms of the precedence and inclusion relations given by the model:

(65) $$[\![\alpha < \beta]\!]^{M,g,c} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g,c} < [\![\beta]\!]^{M,g,c} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

(66) $$[\![\alpha \subseteq \beta]\!]^{M,g,c} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g,c} \subseteq [\![\beta]\!]^{M,g,c} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

The special indexical constant now is interpreted relative to a

[7] As Dowty (1977) discusses, this type of model also offers a simple account imperfective paradox; if the progressive signals that the verbal description holds at a time interval containing the reference time, and the time interval need not be contained in the unique timeline leading up to the moment of speech, then a progressive sentence does not entail that the verbal description was realized. But Dowty (1977) also argues that even under such an account, it is impossible to escape the need for a total ordering among moments. Without it, there would be no way to make sense of counterfactual statements such as *If I were in New York right now, I would do such-and-such* (see pp. 62–66).

given context of utterance c:

(67) $[\![\mathsf{now}]\!]^{M,g,c} = t(c)$

Hence $\mathsf{ill}_{\mathsf{now}}(x)$ is true relative to model M, assignment function g, and context of utterance c if and only if, according to the interpretation function I determined by model M, the ill relation holds between $t(c)$ and $g(x)$:

(68) $[\![\mathsf{ill}_{\mathsf{now}}(x)]\!]^{M,g,c} = \top$ iff $I(\mathsf{ill})(t(c))(g(x))$

11.4 Summary and outlook

This chapter has provided a brief introduction to the semantics of tense and aspect. We argued that tenses can be indexical and anaphoric, and have introduced reference to times in our representation language. We have adopted a two-layer view of aspect: *inclusion aspect* (PROG, PFV) encodes containment relations between the event time and the reference time, while *ordering aspect* (PERF, NPERF) encodes the temporal ordering between the reference time and the event time window. The overt perfect morpheme *have* heads the ordering aspect projection with PERF (event time precedes reference time); the covert NPERF is the default ordering aspect, serving as the identity function: the event time is simultaneous with the reference time. We have also analyzed the future as a prospective aspect operator (WOLL).

We have left a number of important issues unresolved. Some of these have been noted along with way. One phenomenon that we have not covered at all is so-called 'sequence of tense'. Examples include the following:

(69) John decided a week ago that in ten days he would say to his mother that they **were** having their last meal together. (Abusch, 1988)

(70) John said he would buy a fish that **was** still alive.

(Ogihara, 1989)

(71) Mary predicted that she would know that she **was** pregnant the minute she **got** pregnant.
(Kratzer, 1998)

In each of these examples, the bolded phrase is morphologically past tense, but is not interpreted as such. Several authors, starting with Ogihara (1989), have suggested that the tense feature is not semantically interpreted, and that the tense is interpreted as a bound variable. See von Stechow & Grønn (2013a,b) for a recent overview of the discussion. Even more recently, the conversation has been extended to include optional tense languages such as Washo (Bochnak, 2016) and Tlingit (Cable, 2017), in which the hypothesized LF structure is what surfaces in the language.

11.5 Toy fragment

The following extends the toy fragment from Chapter 8 with the additions from this chapter.

Representation language
This chapter extends L_∂ to L_T, a partial, context-sensitive lambda calculus with times. L_T adds a primitive type i for time intervals and two new formula-forming rules:

Syntax rule: Precedence ($<$)
If α and β are expressions of type i, then $\alpha < \beta$ is an expression of type t.

Semantic rule: Precedence ($<$)
$$[\![\alpha < \beta]\!]^{M,g,c} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g,c} < [\![\beta]\!]^{M,g,c} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Syntax rule: Containment ($\subseteq$)
If α and β are expressions of type i, then $\alpha \subseteq \beta$ is an expression of type t.

Semantic rule: Containment ($\subseteq$)
$$[\![\alpha \subseteq \beta]\!]^{M,g,c} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g,c} \subseteq [\![\beta]\!]^{M,g,c} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

We write $\alpha \leq \beta$ as shorthand for $\alpha < \beta \vee \alpha = \beta$. Semantic evaluation is extended to include a context of utterance c, writing $[\![\alpha]\!]^{M,g,c}$. The indexical constants take their values from the context:

- $[\![\mathsf{i}]\!]^{M,g,c} = \mathsf{sp}(c)$ (type e: speaker)
- $[\![\mathsf{u}]\!]^{M,g,c} = \mathsf{ad}(c)$ (type e: addressee)
- $[\![\mathsf{now}]\!]^{M,g,c} = \mathsf{time}(c)$ (type i: speech time)
- $[\![\mathsf{here}]\!]^{M,g,c} = \mathsf{loc}(c)$ (type e: location)

All rules of L_∂ carry over.

Type conventions

All type conventions from Chapter 8 carry over. The following are added:

- Variables of type i: t, t', t'', and indexed variants t_n
- Variables of type $\langle i, t\rangle$: T, U
- Indexical constants of type i: now (speech time); also $\mathsf{5pm}$ (non-indexical)

- Indexical constants of type e: i (speaker), u (addressee), here (location)

Subscript convention: $W_t(x)$ abbreviates $W(x)(t)$; $V_t(x,y)$ abbreviates $V(y)(x)(t)$, where the arguments to V are listed agent-first.

Syntax
All syntax rules from Chapter 8 carry over. The following are added:

$$\begin{array}{lcl} \text{TP} & \rightarrow & \text{T}' \ (\text{Asp}_{\text{Ord}}\text{P} \mid \text{ModP}) \\ \text{T}' & \rightarrow & \text{TENSE T}_n \\ \text{Asp}_{\text{Ord}}\text{P} & \rightarrow & \text{Asp}_{\text{Ord}} \ \text{Asp}_{\text{Inc}}\text{P} \\ \text{Asp}_{\text{Inc}}\text{P} & \rightarrow & \text{Asp}_{\text{Inc}} \ \text{vP} \\ \text{ModP} & \rightarrow & \text{Mod} \ (\text{Asp}_{\text{Ord}}\text{P} \mid \text{Asp}_{\text{Inc}}\text{P}) \end{array}$$

Verbs now take a time argument: intransitive verbs have type $\langle e, \langle i, t\rangle\rangle$; transitive verbs have type $\langle e, \langle e, \langle i, t\rangle\rangle\rangle$.

Composition rules All composition rules from Chapter 8 carry over. All tense and aspect morphemes compose via Function Application.

Lexical entries

V (updated types)
Intransitive verbs follow $W \rightsquigarrow \lambda x.\lambda t.W_t(x)$ (type $\langle e, \langle i, t\rangle\rangle$); transitive verbs follow $V \rightsquigarrow \lambda y.\lambda x.\lambda t.V_t(x,y)$ (type $\langle e, \langle e, \langle i, t\rangle\rangle\rangle$). All verbs from Chapter 8 carry over with this updated type. New verbs:

- *votes*/ *vote*/ *voted*, *dances*/ *dance*/ *danced* follow the intransitive schema.
- *mails*/ *mail*/ *mailed* follows the transitive schema.

Adjectives, nouns, determiners, prepositions, and all other categories from Chapter 8 carry over without change.

$\mathbf{T}_n$ (temporal pronoun, type i)

- $T_n \rightsquigarrow t_n$, where $g(n)$ supplies the reference time. An overt temporal modifier such as *at 5pm* provides this value directly; otherwise it is resolved anaphorically.

TENSE (type $\langle i, i\rangle$)

- PAST $\rightsquigarrow \lambda t\,.\,t^{\mathsf{past}}$
- PRES $\rightsquigarrow \lambda t\,.\,t^{\mathsf{pres}}$

$\mathbf{Asp}_{\mathrm{Ord}}$ (type $\langle\langle i, t\rangle, \langle i, t\rangle\rangle$)

- NPERF $\rightsquigarrow \lambda P_{\langle i,t\rangle}\,.\,\lambda t\,.\,P(t)$
- PERF $\rightsquigarrow \lambda P_{\langle i,t\rangle}\,.\,\lambda t\,.\,\exists t'[t' < t \wedge P(t')]$

$\mathbf{Asp}_{\mathrm{Inc}}$ (type $\langle\langle i, t\rangle, \langle i, t\rangle\rangle$)

- PFV $\rightsquigarrow \lambda P_{\langle i,t\rangle}\,.\,\lambda t\,.\,\exists t''[t'' \subseteq t \wedge P(t'')]$
- PROG $\rightsquigarrow \lambda P_{\langle i,t\rangle}\,.\,\lambda t\,.\,\exists t''[t \subseteq t'' \wedge P(t'')]$

Mod (type $\langle\langle i, t\rangle, \langle i, t\rangle\rangle$)

- WOLL $\rightsquigarrow \lambda P_{\langle i,t\rangle}\,.\,\lambda t\,.\,\exists t'[t < t' \wedge P(t')]$

Syncategorematically interpreted items

All syncategorematically interpreted items from Chapter 8 carry over.

12 | Intensional semantics

12.1 Introduction

12.1.1 Modal flavor and strength

Imagine you're trying to determine if a certain mathematical statement is true or false. After a while, you come to believe that the statement is in fact true, and you even come up with a proof. You think your proof is *probably* correct, but there's a chance you've made a mistake. So you might say to yourself:

(1) It's possible that the statement I'm trying to prove is true, and it's possible that it's false.

In a sense, this is correct: you don't know for sure if the statement is true or false. It might turn out either way. But in another sense, this is incorrect. Mathematical statements are true necessarily if they are true at all. Otherwise, they are necessarily false. The statement you're trying to prove is no different. Even though you don't know if it is true, *in itself* it's either necessarily true or necessarily false; in the first case, it's impossible that it's false, in the second case it's impossible that it's true.

This case illustrates two different senses of the word *possible*. The first sense, *possible for all I know*, is called EPISTEMIC (from the Greek word for "knowledge") or SUBJECTIVE; the second sense, *possible in itself*, is called METAPHYSICAL or OBJECTIVE. It is epistemically contingent whether a mathematical statement that

lacks a proof is true or false; if it is in fact true, it is metaphysically necessary for it to be true. These senses are called MODALITIES.

Epistemic modality and metaphysical modality are examples of MODAL FLAVORS. Other flavors include DEONTIC modality, which relates to rules and norms as determined by some source of authority such as a parent, school policy, societal custom, or law. With deontic modality, possibility corresponds to permission and necessity to obligation. BOULETIC modality has to do with the desires of a given agent; the verb *want* expresses bouletic modality. (See e.g. von Fintel 2006.)

In English, many modal expressions are ambiguous between different modal flavors, and are disambiguated by tense, context, and other factors.

(2) a. There may/might/must be a fly in this room. *epistemic*
b. The coin might/could/would have landed on tails. *metaphysical*
c. Visitors may/must check in at the front desk. *deontic*

While it is not easy to pin down the concepts of metaphysical possibility and necessity, there is one clear way in which they can be distinguished from epistemic possibility and necessity. The epistemic modalities are *subjective* in the sense that they depend on people's knowledge, and different people can know different things. For example, if I know that there is a fly in this room but you don't know if it's a fly or a mosquito, then *There is a fly in this room* is epistemically necessary for me but merely epistemically possible for you. By contrast, the metaphysical modalities are *objective*: for example, whether a coin that actually lands on heads could have landed on tails instead depends on whether the coin is rigged (and whether the world is deterministic) but not on what anyone knows about the coin.

Flavor is not the only dimension along which modals can vary. Another is STRENGTH (or FORCE). Necessity modals like *need* and *must* are STRONG modals, because they express necessity, while

possibility modals like *may* and *can* are WEAK modals, expressing possibility. The logic of possibility and necessity is still in play even when we take flavor into account. For instance, strong and weak modals relate to each other as duals regardless of their flavor; to take a deontic example, *You must not* ϕ is equivalent to *It is not allowed to* ϕ.

The contrasts among the various modal flavors and strengths are among the challenges that we have in developing a theory of modality in natural language. This chapter will lay out some additional challenges, and develop a framework for the analysis of modal phenomena that is designed to meet them.

12.1.2 Necessary vs. contingent truth and falsity

The origin of the term 'modality' has to do with ways (or 'modes') of being true. Two true sentences can be true in different ways. For instance, both of the following sentences are true.

(3) a. Ruth Bader Ginsburg is the first person to be on both *Harvard Law Review* and *Columbia Law Review*.
b. Ruth Bader Ginsburg is Ruth Bader Ginsburg.

But they are true in very different ways. (3a) expresses something which might well have turned out to be false. (3b) expresses something that would still have been true even if things had been very different; it could not have failed to be true. To put it differently, the first expresses a CONTINGENT proposition; the second expresses a NECESSARY one. These are metaphysical, or objective forms of contingency and necessity; they hold regardless of anyone's epistemic state. (As usual, by PROPOSITION, we mean what we meant in Chapter 2: the abstract content expressed by a declarative sentence, independent of the specific words or language used to convey it. Like sentences, propositions can be true or false; unlike sentences, they are language-independent objects. For example, two people who speak different languages might express the same

proposition using different sentences.)

Reflecting the fact that (3a) is contingent and (3b) is necessary, they give rise to claims of different truth status when they are embedded under *necessarily*:

(4) a. Ruth Bader Ginsburg is **necessarily** the first person to be on both *Harvard Law Review* and *Columbia Law Review*. (False)
b. Ruth Bader Ginsburg is **necessarily** Ruth Bader Ginsburg. (True)

Similarly, the negation of (3a) is possible while the negation of (3b) is not:

(5) a. Ruth Bader Ginsburg **might not have been** the first person to be on both *Harvard Law Review* and *Columbia Law Review*. (True)
b. Ruth Bader Ginsburg **might not have been** Ruth Bader Ginsburg. (False)

Words like *necessarily* and *might* are thus sensitive to the distinction between necessary and contingent true propositions.

Contingence and necessity are MODAL properties of propositions. MODALITY refers to ways language can express various relationships that propositions bear to truth. Modality is expressed by adjectives like *necessary*, *possible*, and *contingent*; adverbs like *necessarily*, *possibly*, and *maybe*; auxiliaries like *must*, *may*, *might*, *can*, and *could*; and many more expressions. This chapter develops a formal treatment of modal language.

Exercise 1. Give an example of two *contingent* sentences of English that have the same truth value in reality but express different propositions.

Exercise 2. Like true sentences, false sentences can be either CONTINGENTLY FALSE (false in the actual world but true in other possible worlds) or NECESSARILY FALSE (false in all possible worlds). Give an example of two false sentences, one which is necessarily false, and one which is merely contingently false.

12.1.3 Intension vs. extension and substitutability

The examples we've just been considering illustrate a difference in meaning between these two noun phrases:

(6) a. the first person to serve on both *Harvard Law Review* and *Columbia Law Review*
b. Ruth Bader Ginsburg

How can these two noun phrases have different meanings, if they both refer to Ruth Bader Ginsburg? We can make sense of this by distinguishing between INTENSION and EXTENSION, concepts that we will explain shortly.

In the actual world, the two noun phrases in (6) refer to the same individual, RBG for short. But imagine a possible world in which the forces of patriarchy are just a little bit stronger, and a man is the first to serve on both these law reviews. Relative to that world, the name in (6b) still refers to RBG (she is still the same *person*), but the definite description in (6a) has a different referent.

Relative to a particular world, the *extension* of a name or definite description is a particular individual. The *intension* of such a DP is commonly thought of as a function from possible worlds to individuals. A function from possible worlds to individuals is called an INDIVIDUAL CONCEPT. The intension of (6a) is a partial function which maps each world to whichever woman, if any, was the first to serve on both law reviews *in that world*. The intension of (6b) maps every possible world to RBG, including worlds

in which she does not exist. This latter kind of function is called a RIGID DESIGNATOR; Kripke (1979) famously argued that proper names denote rigid designators. (Letting the name pick out its bearer even at worlds where the bearer does not exist keeps the function total, which we will rely on below when we consider the necessity of identity.) On this view, the two noun phrases in (6) have different intensions, although they are coextensional.

Frege (1892) observed that the logic of natural language does not always adhere to the following principle:

(7) **Substitutability of coextensional expressions**
If two expressions have the same extension, then if one is substituted for the other in any given sentence, the truth value of the sentence remains the same.

This principle is very useful in mathematics; for example, 6*(3+2)=30 and 6*5=30 have the same truth value because (3+2) and 5 have the same extension.

The semantics we have developed so far has also adhered to this principle. And indeed, in many cases it is safe to follow. For example, the following argument is valid:

(8) a. Ruth Bader Ginsburg was the first person to be on both *Harvard Law Review* and *Columbia Law Review*.
b. No law firm in New York City hired Ruth Bader Ginsburg after she finished law school.
c. Therefore, no law firm in New York City hired the first person to be on both *Harvard Law Review* and *Columbia Law Review* after she finished law school.

The conclusion is licensed by the fact that Ruth Bader Ginsburg *is* the first person to be on both *Harvard Law Review* and *Columbia Law Review* – these two noun phrases denote the same person, so they are coextensional.

But there are examples where the principle of substitutability of coextensionals does not hold. We already saw one instance of

this above, with examples (5a) and (5b), repeated here:

(9) a. RBG might not have been the first person to serve on the *Harvard Law Review* and the *Columbia Law Review*.
b. RBG might not have been RBG.

To convince your neighbor that (9a) is true, you could simply point out that there was nothing inevitable about Ruth Bader Ginsburg's career. If she had decided to become, say, a postal worker, she might well not have worked on any law journals. There is a natural reading of (9a), perhaps its most prominent one, which seems true for this reason. By contrast, even if she had become a postal worker, she would still have been herself; even if she had had a different *name*, she would still have been herself; in this sense, (9b) is clearly false. (Here, we are focusing on the reading of (9b) that is similar to *RBG might have been someone else* or *RBG might have been distinct from herself*. To the extent that (9b) can also be used to express that RBG might have had a different name, we set that reading aside.) Indeed, it seems hard to imagine what it would even take for (9b) to be true. Accordingly, following Kripke (1980), it is commonly held that identity is a metaphysically necessary relation. No individual could fail to be identical to itself, even under vastly different circumstances. (The kind of identity we are talking about here is NUMERICAL IDENTITY: it is the relation that each thing bears only to itself. It is not about self-conception, social presentation, or any other properties that make a thing or person unique.)

Environments in which the principle of substitutability of coextensionals fails are called OPAQUE. As we have seen, modals produce opaque environments. Environments where the principle of substitutability of coextensionals succeeds are called TRANSPARENT. An example of a transparent environment is negation, as you may recall, is a 'truth-functional' connective. In other words, the truth value of the negation of a sentence depends only on the *ex-*

tension of the sentence being negated. To put it in yet another way, negation is an extensional operator, in contrast to modals, which are intensional operators. Among the challenges in developing a theory of modality is to explain the failure of substitutability of co-extensionals in opaque environments.

Consider another example, used by Quine (1951) in a discussion of the difference between intension and extension. It just so happens that (so far as we know) every species that has a heart also has a kidney. In the actual world, *creature with a heart* picks out the same set that *creature with a kidney* does. In this sense, these two expressions are COEXTENSIONAL, i.e., they have the same EXTENSION (in the actual world; we will drop this qualification from here on in). But it could have been otherwise; there are other possible worlds where there are creatures that have hearts but no kidneys, or *vice versa*. So at some level, these two expressions do not have the same *meaning*, even though relative to the actual world, they pick out the same set of individuals. And they are not COINTENSIONAL; they do not have the same intension. Another way of putting it is that they are not INTENSIONALLY EQUIVALENT.

Intension and extension can be thought of as two different semantic values that an expression has. Observe that *intension* is spelled with an 's'; this crucial letter distinguishes it from the entirely separate concept of *intention* with a 't'. The intension of an expression like *creature with a heart* can be formally represented as a function that takes as input a world w and returns the extension of the expression at that world.

Sentences, too, have both intensions and extensions. Following in Frege's footsteps, as is common, we take the extension of a sentence to be a truth value, such as T or F. It follows that any two true sentences are coextensional. (The same goes for any two false sentences.) For instance, (3a) and (3b) are coextensional, because they are both true.

Clearly, the extension of a sentence cannot be its meaning; otherwise (3a) and (3b) and all other true sentences would have

the same meaning (and so would all false sentences). A common approach is to say that sentence meanings are propositions, and that two sentences have the same meaning just in case they express the same proposition. It is common to assume that propositions are the kinds of things that one believes, knows, doubts, etc. Since someone might doubt (3a) without doubting (3b), or might believe (3b) without believing (3a), these two sentences must express different propositions.

Formally, it is common to model a proposition as a set of possible worlds. The proposition expressed by a sentence (and more generally by any sentential clause, embedded or not) is the set of possible worlds in which the sentence is true. The intension of a sentence can be seen as the proposition it expresses, or the characteristic function thereof, that is, a function from possible worlds to truth values.

The intension of a necessary sentence maps every possible world to the truth value T; the intension of a contingent sentence maps some possible worlds to T and others to F. Since (3a) is contingent and (3b) is necessary, they do not have the same intension, even though they are coextensional. The contrasts between those two sentences with respect to modals like *necessarily* and *might* can be traced back to these differences at the intensional level.

The contribution of modal words like *might* and *necessarily* to the meaning of a sentence evidently depends not (merely) on the extension of the constituent they attach to, but on its intension. This is true of verbs like *believe*, *want*, and *seek*, as well. Montague's (1974b) Intensional Logic (IL) is a formal representation language that allows for reference to intensions through a special device called the 'hat operator', as we will explain later. The system that we will develop here is heavily influenced by Montague's IL, but the exact formalism we will ultimately put forward adheres to Gallin's (1975) Ty2, a logic that includes explicit quantification over possible worlds.

Exercise 3. Give an example of two *distinct* sentences of English that are cointensional. In addition, specify whether they are necessarily true, necessarily false, or contingent.

12.2 Necessity and possibility

In this section, we lay out some basic desiderata for a theory of necessity and possibility. After that, we will build up a theory that captures these facts. We will discuss further desiderata for a theory of modality and how to meet them later in the chapter.

12.2.1 Necessity and possibility are duals

The logic of necessity and possibility mirrors the logic of universal and existential quantification. The negation of a necessity statement like (10a) is equivalent to a possibility statement about a negation as in (10b).

(10) a. It's **not necessary** for you to take out the trash.
b. It's **possible** for you **not** to take out the trash.

Compare (10) to (11).

(11) a. **Not everybody** took out the trash.
b. **Someone** did **not** take out the trash.

Just as the negation of a necessity statement about ϕ is equivalent to a possibility statement about the negation of ϕ, the negation of a universal is equivalent to an existential negative.

Similarly, to the negate the possibility of a negation is to state the necessity of what is negated:

(12) a. It's **not possible** for me **not** to pay the fine.
b. It's **necessary** for me to pay the fine.

Similarly, negating the existence of someone who does not satisfy a property is equivalent to a universal attribution of the property. Compare (12) to (13):

(13) a. It's **not** the case that **someone** did **not** pay the fine.
b. **Everybody** paid the fine.

These equivalences are characteristic of pairs of operators that are DUALS of each other. $\forall$ and $\exists$ are duals of each other because $\forall x.\phi$ is equivalent to $\neg\exists x.\neg\phi$ and $\exists x.\phi$ is equivalent to $\neg\forall x.\neg\phi$. Just as universal and existential quantifiers are duals of each other, so are necessity and possibility: 'necessarily' is equivalent to 'not possibly not', and 'possibly' is equivalent to 'not necessarily not'.

These patterns can be understood under a view where necessity involves universal quantification and possibility involves existential quantification. The difference is that in the case of necessity and possibility, it is possible worlds rather than individuals that are being quantified over.

12.2.2 Veridicality

Another valid argument involving modal operators involves the inference from necessary truth to simple truth:

(14) a. Necessarily, this bottle is made of plastic.
b. ∴ This bottle is made of plastic.

From possible truth, of course, we cannot get to simple truth:

(15) a. Possibly, this bottle is made of plastic.
b. #∴ This bottle is made of plastic.

This illustrates that the modal adverb *necessarily*, unlike *possibly*, is VERIDICAL; that is, *necessarily* entails the truth of its propositional argument.

An analogy with universal and existential quantification can be made here, too:

(16) a. Every bottle is made of plastic.
b. ∴ This bottle is made of plastic.

(17) a. Some bottle is made of plastic.
b. ∴ This bottle is made of plastic.

Here, too, the logic of necessity and possibility mirrors the logic of universal and existential quantification.

12.3 Toward a theory of modality

12.3.1 Modal logic

MODAL LOGICS are formal languages in which it is possible to make statements about necessity and possibility. The most widespread type of semantics for modal logics, POSSIBLE-WORLD SEMANTICS, provides an elegant account of this kind of reasoning. The idea is to treat *necessarily* and *possibly* as universal and existential quantifiers over possible worlds respectively, in analogy to the definitions of necessary and possible propositions mentioned above. A proposition is then deemed to be NECESSARY just in case it is true in all relevant possible worlds; and a proposition is POSSIBLE just in case it is true in at least one relevant possible world.

In PROPOSITIONAL MODAL LOGIC, where there is no reference to or quantification over individuals, the basic expressions are proposition letters like p, q, and r, just as in propositional logic. Models for propositional modal logic determine a set of possible worlds W and an interpretation function I that specify which proposition letters are true in which worlds. For example, there could be a model M_1 which contains a set W_1 of possible worlds. Suppose that W_1 contains just four possible worlds, which we will call World A, World B, World C, and World D. This model M_1 might have an interpretation function I_1 specifying that p is true in World A and World B but not in World C and World D. The term KRIPKE MODEL (or KRIPKE FRAME) can be used to describe this sort of

model, where possible worlds are included (Kripke, 1963).

Models for modal logic also specify an ACCESSIBILITY RELATION among possible worlds, written $\mathcal{R}$. For example, M_1 might specify the set of worlds that are accessible from World A include itself (World A), along with World B and World C, but not World D. The necessity operator $\Box$ 'box' and the possibility operator $\Diamond$ 'diamond' have their semantics defined in terms of the accessibility relation:

- $\Box\phi$ is true relative to M and w just in case ϕ is true in *every* possible world w' that is accessible from w according to M;
- $\Diamond\phi$ is true relative to M and w just in case ϕ is true in *some* possible world w' that is accessible from w according to M.

More formally, if M furnishes the accessibility relation $\mathcal{R}$:

(18) In modal logic:
   a. $[\![\Box\phi]\!]^{M,w} = \mathsf{T}$ just in case $[\![\phi]\!]^{M,w'} = \mathsf{T}$ for *every* possible world w' such that $w\mathcal{R}w'$.
   b. $[\![\Diamond\phi]\!]^{M,w} = \mathsf{T}$ just in case $[\![\phi]\!]^{M,w'} = \mathsf{T}$ for *some* possible world w' such that $w\mathcal{R}w'$.

In the simplest case, there is only one accessiblity relation $\mathcal{R}$ among worlds, and it is transitive, symmetric, and reflexive (see Hughes & Cresswell 1968 for a fuller introduction; von Fintel & Heim 2011 also give a helpful pedagogical discussion of these concepts). Different patterns of logical inference arise under different assumptions about the properties of the accessibility relation. For instance, if the accessibility relation is reflexive (so any given world is always accessible to itself), then it can be proven that $\Box\phi$ entails ϕ, for any formula ϕ. That is, for any given model M and any given world w, if $\Box\phi$ is true relative to M and w, then so is ϕ, as long as the accessibility relation in M is reflexive. The correspondence in fact holds in both directions: $\Box\phi$ entails ϕ for every formula ϕ if and only if the accessibility relation is reflexive.

Let us continue to assume that M_1 has an interpretation function I_1 specifying that p is true in World A and World B but not in World C and World D. Suppose further that in M_1, every world is accessible from every other world. Let's consider some different formulas and consider what truth values they have relative to various worlds.

(19) In modal logic, given M_1 as specified above:
$[\![\mathsf{p}]\!]^{M_1,\mathrm{A}} = \mathsf{T}$ $[\![\Box\mathsf{p}]\!]^{M_1,\mathrm{A}} = \mathsf{F}$ $[\![\Diamond\mathsf{p}]\!]^{M_1,\mathrm{A}} = \mathsf{T}$
$[\![\mathsf{p}]\!]^{M_1,\mathrm{C}} = \mathsf{F}$ $[\![\Box\mathsf{p}]\!]^{M_1,\mathrm{C}} = \mathsf{F}$ $[\![\Diamond\mathsf{p}]\!]^{M_1,\mathrm{C}} = \mathsf{T}$

Even though p is not true relative to World C, $\Diamond$p is, because there is at least one world that is accessible from World C, for example World A, where p is true.

Exercise 4. Add rows to the table in (19) for World B and World D.

Under this semantics, $\Box$ and $\Diamond$ are, essentially, quantifiers over possible worlds. It follows that there are many parallels between $\Box$ and $\forall$, and between $\Diamond$ and $\exists$. For example,

$$\neg \Diamond \neg \phi$$

and

$$\Box \phi$$

are equivalent, and this can be used to explain the equivalence of the English sentences (12a) and (12b). In modal logic, generally, $\Box$ and $\Diamond$ are duals of each other, just like $\forall$ and $\exists$ are in predicate logic.

12.3.2 Explicit reference to worlds

It is possible to simulate modal logic in predicate logic by bringing out the hidden quantification over possible worlds into the open,

binding them with ordinary quantifiers, and dispensing with the boxes and diamonds. To this end, we introduce variables w, w', w'', etc, that range over possible worlds. Let p be a predicate of worlds that holds of a world w if Anna pays a fine in w. Then when Anna utters (12b) (*It's necessary for me to pay a fine*), the content of her utterance can be captured with universal quantification over possible worlds, as in the following formula:

(20) $\forall w . \mathsf{p}(w)$

Ty2 is a logic that has this sort of quantification over possible worlds. The name of the language derives from the fact that there are two basic types other than t, e for individuals and s for possible worlds. Expressions of type s denote particular possible worlds; expressions of, for example, type $\langle s, t \rangle$ denote functions from possible worlds to truth values. In Ty2, the formula in (20) is assigned a truth value relative to a model M, which furnishes a set of possible worlds W, just as in modal logic. An interpretation function I assigns a value to all non-logical constants, such as p. This particular constant is of type $\langle s, t \rangle$, as it denotes a function from possible worlds to truth values.

The equivalent sentence invoking possibility ((12a); *It's not possible for me to not pay a fine*) can be represented using existential quantification:

(21) $\neg \exists w . \neg \mathsf{p}(w)$

In words: there is no world that is not a p-world. The equivalence of (20) and (21) reflects the fact that $\forall$ and $\exists$ are duals.

Now, in modal logic, truth is relative to a model M and a world w. The world relative to which truth is evaluated is called the EVALUATION WORLD. A contingent statement is one whose truth depends on the choice of evaluation world. In Ty2, contingent statements are represented using formulas that contain a free variable for the evaluation world (Gallin, 1975). We use the special world variable w_0 to designate the evaluation world. Thus rather

than writing (22a) we write (22b):

(22) a. $[\![\mathsf{p}]\!]^{M,w} = \mathsf{T}$ (modal logic style)
b. $[\![\mathsf{p}(w_0)]\!]^{M,g} = \mathsf{T}$ (Ty2 style)

Whether or not (22b) is true depends on which value is assigned to the special world variable w_0 by the assignment function g.

Recall these two definitions from modal logic:

(23) a. $[\![\Box\phi]\!]^{M,w} = \mathsf{T}$ just in case $[\![\phi]\!]^{M,w'} = \mathsf{T}$ for *every* possible world w' such that $w\mathcal{R}w'$.
b. $[\![\Diamond\phi]\!]^{M,w} = \mathsf{T}$ just in case $[\![\phi]\!]^{M,w'} = \mathsf{T}$ for *some* possible world w' such that $w\mathcal{R}w'$.

The box of modal logic ($\Box$) can be simulated in Ty2 by replacing (24a) with (24b):

(24) a. $[\![\Box\mathsf{p}]\!]^{M,w} = \mathsf{T}$ (modal logic style)
b. $[\![\forall w.[\mathcal{R}_{w_0}(w) \rightarrow \mathsf{p}(w)]]\!]^{M,g} = \mathsf{T}$ (Ty2 style)

If p holds in every world that is accessible from the world of evaluation, then the formula is true.

Likewise, in place of modal logic's (25a), we can use (25b) in Ty2 in order to simulate the diamond in modal logic ($\Diamond$).

(25) a. $[\![\Diamond\mathsf{p}]\!]^{M,w} = \mathsf{T}$ (modal logic style)
b. $[\![\exists w.[\mathcal{R}_{w_0}(w) \wedge \mathsf{p}(w)]]\!]^{M} = \mathsf{T}$ (Ty2 style)

This formula requires that p holds in some world that is accessible from the world of evaluation.

Let q be a constant of type $\langle s,t\rangle$ denoting the proposition that Alex takes out the trash. Suppose that the accessibility relation is deontic, holding only between deontically accessible worlds, hence the deontic flavor of the modality. Then *It's not necessary for Alex to take out the trash* can be represented in Ty2 as follows:

(26) $\neg\forall w.[\mathcal{R}_{w_0}(w) \rightarrow \mathsf{q}(w)]$

Similarly, *It's possible for Alex not to take out the trash* can be represented:

(27) $\exists w.[\mathcal{R}_{w_0}(w) \wedge \neg \mathsf{q}(w)]$

These formulas are equivalent, as we know from our prior study of universal and existential quantification.

For simple cases like this equivalence, modal logic and its simulation in predicate logic amount to the same thing. But there are things one can express with explicit quantification over worlds that are impossible to express in modal logic. In other words, exposing world variables and making them accessible to quantifiers provides the means to account for contrasts that are beyond the reach of the boxes and diamonds of modal logic. The following is based on a classical example due to Cresswell (2012):

(28) It might have been that everyone rich was poor.

One reading of this sentence is nonsensical. It says that things could have been different in such a way that being rich entails being poor. In modal logic (enriched with Predicate Logic):

(29) $\Diamond \forall x[\mathsf{rich}(x) \rightarrow \mathsf{poor}(x)]$

Another reading, which is sometimes paraphrased as *It might have been that everyone who is in fact (or: actually) rich was poor*, says that things could have been different in such a way that everybody who is rich as things stand in the actual world would in that case have been poor. Or to put it differently, it says that things could have been turned out in such a way that none of the people who really are rich would have been rich. This is true, for example, if you believe that it would be possible for every rich person to simultaneously give all their money to a poor person, or if you believe that the global world economy to tank in such a way that everyone becomes poor.

But this reading can't be expressed using boxes and diamonds.

In particular, this won't work:

(30) $\forall x[\mathsf{rich}(x) \rightarrow \Diamond \mathsf{poor}(x)]$

This says that for every rich person, there is a possibility that that person could have been poor. In other words, nobody who is rich is rich of necessity. This can be true even if it would have been impossible for everyone to be poor at the same time.

But suppose that the meaning of *could* is represented using existential quantification over possible worlds. Then the relevant reading can be captured by the following formula:[1]

(31) $\exists w.\forall x.[\mathsf{rich}(@)(x) \rightarrow \mathsf{poor}(w)(x)]$

Suppose that there is one possible world that is the actual world, and we can refer to it in the representation language using $@$. Then $\mathsf{rich}(@)(x)$ expresses the idea that x is rich in the actual world, or that x is "actually" rich. Likewise, $\mathsf{rich}(w)(x)$ expresses that x is rich in w. Here, w is a variable over possible worlds that is bound by an existential quantifier; the formula says that there is a world w such that everyone who is rich in the actual world poor in w. This is correct. So, with the ability to make reference to possible worlds in the representation language, we can provide an adequate account of the truth conditions for this possibility statement.

12.3.3 Back to substitution failures

Let us return to the examples we started with, and consider how their truth conditions could be represented in Ty2. Suppose that

[1] We are glossing over the fact that this formula does not make reference to a world of evaluation. A more careful treatment would do so, and add as a constraint on w that it be accessible from the world of evaluation:

$$\exists w.[\mathcal{R}_{w_0}(w) \wedge \forall x.[\mathsf{rich}_{@}(x) \rightarrow \mathsf{poor}_{w}(x)]]$$

rbg is a constant of type e that denotes a particular individual (RBG). Suppose that we use the constant first to denote a function that, given a possible world and an individual, yields 'true' if that individual was first to serve on both the *Columbia Law Review* and the *Harvard Law Review* in that world. This constant has type $\langle s, \langle e, t \rangle \rangle$. We write its world argument using a subscript, so

$$\mathsf{first}_w(x)$$

means that x was first to serve on both law reviews in world w. The claim that Ruth Bader Ginsburg was the first to do these things can be represented in Ty2 as follows:

(32) $\mathsf{rbg} = \iota x . \mathsf{first}_{w_0}(x)$

The formula in (32) is true in the evaluation world if the unique individual who was first to serve on both of these law reviews *in that world* is Ruth Bader Ginsburg. The truth of this formula could therefore vary from evaluation world to evaluation world. In other words, this formula is *contingent*, just like the corresponding English sentence. The claim that Ruth Bader Ginsburg is numerically identical to herself, on the other hand, can be represented as follows:

(33) $\mathsf{rbg} = \mathsf{rbg}$

Here there is not even any reference to the evaluation world, so its truth clearly does not depend on one. Regardless of our choice of value for the variable w_0, the statement will be true. In this sense, the formula is necessarily true.

Formulas that don't make any reference to an evaluation world are not contingent, but there are necessary formulas that do contain reference to an evaluation world. To illustrate, consider the name *Miss America*. The denotation of this name can in principle vary from world to world. Let us introduce a constant ma which denotes a function from possible worlds to the individual who is Miss America in that world (if any). The sentence *Miss America*

is Miss America is necessarily true, even though the identity of Miss America can vary from world to world. We can represent the meaning of that sentence in Ty2 as follows:

(34) $\mathsf{ma}_{w_0} = \mathsf{ma}_{w_0}$

This is also a necessary statement, because the function denoted by ma will yield the same individual on the left-hand and right-hand sides of the equation, regardless of the value of w_0.

The claim that RBG was *necessarily* the first person to serve on both the Harvard Law Review and the Columbia Law Review can be represented in Ty2 as follows:

(35) $\forall w[\mathcal{R}_{w_0}(w) \rightarrow \mathsf{rbg} = \iota x.\mathsf{first}_w(x)]$

This says that in every world accessible from the evaluation world, the individual who was first to serve on both of these law reviews was Ruth Bader Ginsburg. Given the existence of worlds where this does not hold, this particular necessity claim is correctly predicted to be false.

The sentence *Ruth Bader Ginsburg is necessarily Ruth Bader Ginsburg* can be represented in Ty2 as follows:

(36) $\forall w[\mathcal{R}_{w_0}(w) \rightarrow \mathsf{rbg} = \mathsf{rbg}]$

Notice that this has something like vacuous quantification over w, because the formula on the right-hand side of the arrow does not make reference to it. Hence the claim just boils down to the numerical identity between RBG and herself. Hence we correctly derive that the necessity claim is true.

It's possible to have a true necessity claim without the near-vacuous quantification that we have in (36). For example, consider the sentence *Miss America is necessarily Miss America.* The representation in Ty2 for this sentence would be as follows:

(37) $\forall w[\mathcal{R}_{w_0}(w) \rightarrow \mathsf{ma}_{w_0} = \mathsf{ma}_{w_0}]$

Here we do not have anything like vacuous quantification over w, but the necessity claim is clearly true, for the reason explained above.[2]

[2]We have assumed that proper names like *Ruth Bader Ginsburg* pick out the same individual relative to every world, in other words, that they are rigid designators. In this respect our treatment of proper names is not unlike the view associated with 19th century British philosopher John Stuart Mill ('Millianism'), where the denotation of a proper name is just its referent, i.e. the bearer of the name, and that this is all that the name contributes to propositions.

Frege noticed a serious difficulty with Mill's view: statements like *Hesperus is Phosphorus* and *Hesperus is Hesperus* are very different in status. According to Millianism, these two statements denote the same proposition, namely that Venus is Venus. Frege's puzzle, as it is usually known, is the following. The first statement conveys an important piece of information: it was the result of an empirical discovery by Greek astronomers, and it would have been news to the ancient Babylonians. The second statement, by contrast, is entirely uninformative. How can this be, if they denote the same proposition?

Frege's own reaction to this puzzle was to distinguish between the 'sense' (Sinn) and 'reference' (Bedeutung) of an expression. According to Frege, the words *Hesperus* and *Phosphorus* have the same referent (namely Venus), but they have different senses. It's not clear exactly what a sense is, though (a 'mode of presentation', he says). Another approach is to take a name like *Aristotle* to be synonymous with a definite description like *the teacher of Alexander the Great*. This approach is called 'descriptivism' about proper names, because it essentially treats proper names as definite descriptions in disguise. Descriptivism also has its challenges and is not generally considered a live option.

Today, there is no consensus on the best way to solve Frege's puzzle. One approach is worth mentioning; it draws on two-dimensional semantics to capture the distinction between epistemic and metaphysical modality (Chalmers, 2006). On this approach, a proposition, and any expression within it, takes two parameters: a contextual parameter, corresponding to epistemic possibilities, and a possible world, corresponding to metaphysical possibilities. Relative to a context in which "Hesperus" and "Phosphorus" are known to have the same referent, the two noun phrases denote the same individual as each other in every possible world. Relative to a context in which "Hesperus" and "Phosphorus" are believed to have different referents, the two noun phrases denote the same two distinct individuals in every possible world.

12.4 Compositionality with Ty2

In this section, we will work towards a compositional semantics for a fragment of English that contains modal language. As we will argue shortly, one of the criteria for such a system is that expressions be able to combine with the intensions rather than extensions of their syntactic dependents, if not always, at least sometimes.

12.4.1 Selecting for intensions

While (38a) implies that there is at least one sloop (a type of sailboat), (38b) need not do so:

(38) a. Andrea sees a sloop.
b. Andrea wants a sloop.

As Quine (1956) puts it, example (38b) can be interpreted to mean that Andrea merely seeks relief from slooplessness, and not a particular sloop; no sloops need even exist for the sentence to be true. What Andrea wants is to be in a world w where there is some thing x such that x is a sloop in w and Andrea has x in w. In order to specify such truth conditions, it is necessary for *want* to have access not just to the extension of *sloop* in the actual world (which may be the empty set), but to the intension of *sloop*, a function from possible worlds to sets of individuals. In this sense, *want* is an INTENSIONAL VERB, while *see* is an EXTENSIONAL VERB. The existence of intensional verbs shows that we need a theory of composition that allows a head to select for the intension of its complement.

The observations made very early on in this chapter about *necessarily* and *possibly* reinforce this point. The truth *necessarily* ϕ depends not just on whether ϕ is true in the world of evaluation, but on its truth value across all possible worlds. In other words, the semantics of *necessarily* ϕ depends on the intension of ϕ, and not just on its extension at a particular world. This obser-

vation also supports the conclusion that our compositional system should allow operators access to the intensions of their prejacents.

12.4.2 History: the hat operator

One compositional mechanism by which a linguistic expression can select for the intension of one of its syntactic dependents was developed by Montague (1974b). In this work ('PTQ'), Montague gave a compositional fragment of English in which natural languages were translated into a representation language called Intensional Logic (IL). Montague's IL is like modal logic in that it has no explicit reference to possible worlds, but also has all of the richness of lambda calculus in terms of semantic types, so it is possible to compositionally assign truth conditions to sentences of natural language using this representation language.

Among the devices of Montague's IL is the HAT OPERATOR. This is a device in the representation language for referring directly to the intension of an expression. From any expression α, a new expression denoting the intension of α can be formed using the symbol $\hat{}$ ('hat'). The intension of a given expression α is denoted in the representation language as follows:

$$\hat{}\alpha$$

Relative to any given world, this expression has as its *extension* the *intension* of α. Hence for any M, g, and w:

(39) In Montague's IL:
$[\![\hat{}\alpha]\!]^{M,g,w}$ is that function f such that
for all $w' \in W$: $f(w') = [\![\alpha]\!]^{M,g,w'}$.

For example, if α is unicorn then $\hat{}\alpha$ is $\hat{}$unicorn. Suppose that in w_1, there are no unicorns. Then the extension of unicorn in w_1 is the characteristic function of the empty set. The intension of unicorn

is a function that takes as input a world w and returns the extension of unicorn at w. It is this function—the intension of unicorn—that ^unicorn has as its ordinary denotation, i.e., as its extension. So it's a device for pointing at the intension.

12.4.3 Forming the intension

In Ty2, given that we treat w_0 as a special variable designating the world of evaluation, we simulate the effect of Montague's hat operator in a different way. To see this, let us first discuss the relationship between Montague's IL and Gallin's Ty2.

Gallin (1975) actually provided a translation procedure to get from IL to Ty2. There are different translation rules depending on whether the expression is a constant or a variable, etc. For example, assume that unicorn is a constant of type $\langle e, t\rangle$ in Montague's IL. Then there is a corresponding eponymous constant unicorn in Ty2 of type $\langle s, \langle e, t\rangle\rangle$. It denotes a function from possible worlds to (characteristic functions of) sets. The extension of the word *unicorn* relative to the evaluation world is denoted in Ty2 by applying $\langle s, \langle e, t\rangle\rangle$-type unicorn to the evaluation world.

(40)

Type	**Montague's IL**	**Ty2**
$\langle e, t\rangle$	unicorn	unicorn_{w_0}
e	ma	ma_{w_0}

For every constant type τ in IL, there is a corresponding constant in Ty2 of type $\langle s, \tau\rangle$. One consequence of this is that a type e constant of IL has a corresponding constant of type $\langle s, e\rangle$ in Ty2. Strictly speaking, Gallin did not have any constants of type e, all constants were of type $\langle s, \tau\rangle$ for some type τ. Our use of constants of type e is a slight deviation from Gallin's system.

If α is an IL constant, then its translation into Ty2 is α_{w_0}. Thus on the Ty2 side, the intension-denoting constant is given the special evaluation world variable w_0 as an argument.

In unicorn_{w_0}, the evaluation world variable w_0 is free. But w_0 can be bound by a lambda operator, yielding a function that de-

notes the intension. This produces the effect of Montague's hat operator.

(41)

Type	**Montague's IL**	**Ty2**
$\langle s, \langle e, t \rangle \rangle$	ˆunicorn	$\lambda w_0 . \text{unicorn}_{w_0}$
$\langle s, e \rangle$	ˆma	$\lambda w_0 . \text{ma}_{w_0}$

Thus we can simulate the effect of the hat operator simply by prepending λw_0.

12.4.4 Composition rules for an intensional fragment

12.4.4.1 Function Application

We are now ready to implement a compositional mechanism by which a linguistic expression can select for the intension of one of its syntactic dependents. The idea of 'selecting for an intension' comes from Montague, but since we are using a Ty2-like logic that has explicit reference to possible worlds, our solution looks slightly different than Montague's. It looks almost exactly like the one presented by Heim & Kratzer (1998).

The Function Application rule that we use in this system is exactly as before:

Composition Rule. Function Application (FA)
Let γ be a syntax tree whose only two subtrees are α and β (in any order) where:

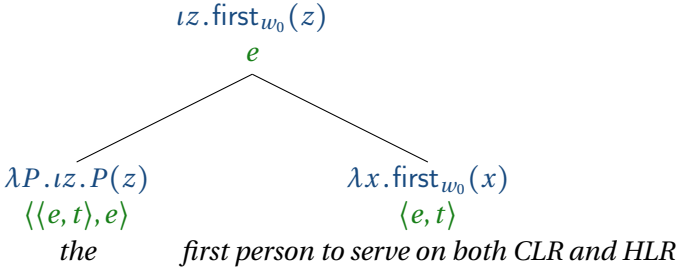
- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type σ.

Then

$$\gamma \rightsquigarrow \alpha'(\beta')$$

Example: Definite descriptions. Suppose that we assume that *first person to serve on both CLR and HLR* is translated into Ty2 as $\lambda x.\mathsf{first}_{w_0}(x)$, an expression of type $\langle e,t\rangle$. Suppose we use an ordinary Fregean treatment of the definite article. Then for *the first person to serve on both CLR and HLR*, FA gives:

(42) Using Function Application (FA):



12.4.4.2 Intensional Function Application

For expressions whose extension at a given world depends not only on the extensions of the parts at that world, but also on their intensions, Heim & Kratzer (1998) propose an additional mode of composition that was inspired by Montague's hat operator and that has come to be known as Intensional Function Application. This rule combines an intension-seeking predicate with the intension of its syntactic complement.

Composition Rule. Intensional Function Application (IFA)
Let γ be a syntax tree whose only two subtrees are α and β (in any order) where:

- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle\langle s,\sigma\rangle,\tau\rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type σ.

Then

$$\gamma \rightsquigarrow \alpha'(\lambda w_0.\beta')$$

Thus Intensional Function Application is a composition rule that combines one daughter's extension with the intension of the other daughter.

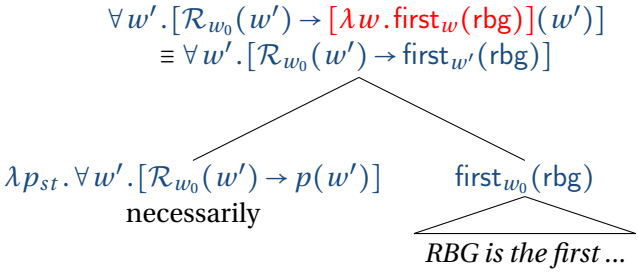
Example: *necessarily*. One case where we will want to use this rule is with modal adverbs like *necessarily*, because their contribution to the meaning depends on the intension of the clause they combine with. Suppose we assume that *necessarily* is translated as follows:

(43) *necessarily* $\rightsquigarrow \lambda p_{st} . \forall w' . [\mathcal{R}_{w_0}(w') \rightarrow p(w')]$
$\langle st, t \rangle$

Then we can use IFA to combine this meaning with the intension of the sentence that the adverb combines with. For example, consider:

(44) Necessarily, RBG is the first person to serve on both the CLR and the HLR.

(45) Using Intensional Function Application (IFA):



As shown in the tree, the extension of *RBG is the first...* is captured by the formula $\mathsf{first}_{w_0}(\mathsf{rbg})$. The intension is $\lambda w_0 . \mathsf{first}_{w_0}(\mathsf{rbg})$, or equivalently, after re-lettering the bound variable, $\lambda w . \mathsf{first}_{w}(\mathsf{rbg})$. The top line of the tree shows that it is this intension that goes in for the $\langle s, t \rangle$-type variable p. This compositional analysis accounts for the fact that the truth of a *necessarily* claim about ϕ depends not just on whether or not ϕ is true, but also on its truth

value in other possible worlds.

Exercise 5. Give a lexical entry for *possibly* on the model of the one for *necessarily* just shown, and derive a meaning for a sentence including it using IFA.

Example: Intensional transitive verbs. Recall the distinction between *see* and *want* discussed above in connection with examples (38a) *Andrea sees a sloop* and (38b) *Andrea wants a sloop*. The former implies the existence of sloops, while the latter does not. In his seminal work on intensional semantics (Montague, 1974b), Montague discussed a similar contrast between *find*, an extensional verb, and *seek*, an intensional verb:

(46) a. Andrea found a unicorn.
b. Andrea sought a unicorn.

Example (46a) implies that there are unicorns; (46b), on one of its readings, does not; to borrow Quine's (1956) phraseology, it just means that Andrea sought relief from unicornlessness. In other words, *find* has existential import and that reading of *seek* does not.

With the tools that we now have in hand, we can model this distinction as follows. An extensional transitive verb like *finds* has a translation of type $\langle e, \langle e, t\rangle\rangle$:

(47) $\textit{finds} \rightsquigarrow \lambda y \lambda x . \mathsf{find}_{w_0}(x, y)$

The translation for *find a unicorn* involves QR of *a unicorn*, giving the LF in (48a) and the Ty2 translation in (48b)

(48) a. LF: [a unicorn] λ_1 [Andrea found t_1]
b. $\exists y[\mathsf{unicorn}_{w_0}(y) \wedge \mathsf{find}_{w_0}(\mathsf{a}, y)]$

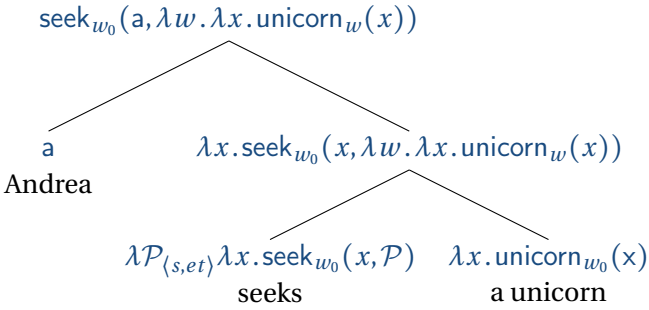
This derivation uses only compositional techniques that are familiar from the extensional systems from previous chapters.

Intensional verbs, on the other hand, expect intensional inputs; in particular, *seeks* expects type $\langle s, et\rangle$.

(49) *seeks* $\rightsquigarrow \lambda\mathcal{P}_{\langle s,et\rangle}\lambda x.\mathsf{seek}_{w_0}(x,\mathcal{P})$

Given that *a unicorn* is an expression of type $\langle e, t\rangle$, it can combine with *seek* via Intensional Function Application like so:

(50)



The formula that is obtained at the top node expresses a relation between an individual and a property. Existence of unicorns is not implied by that formula.

12.5 Modal flavors and attitudes

12.5.1 Modal flavor

To capture modal flavor, one strategy is to adopt multiple different accessibility relations, one for each flavor. Thus a world w' might be epistemically accessible from w but not deontically accessible, etc. Propositional attitudes like belief and desire can be modelled using similar mechanisms, although in these cases the accessibility relation is relativized to a particular agent.

We introduce a predicate constant of type $\langle\langle s, \langle s, t\rangle\rangle, t\rangle$ for each flavor, true of accessibility relations of that kind:

- deon: deontic accessibility (compatible with the rules in force)
- epist: epistemic accessibility (compatible with what is known)
- circ: circumstantial accessibility (reflecting genuine possibility)

There can in principle be more than one accessibility relation of a given flavor. The verb *have to* is unambiguously deontic in flavor, for example, but the relevant set of rules can differ from context to context: in some cases the speaker may be talking about what is legal in the local country; in others, it may be rules of the house. To allow for this kind of flexibility, we represent the meaning of *have to* using a free variable $\mathcal{R}$, of type $\langle s, \langle s, t\rangle\rangle$. We then constrain it to be a deontic accessibility relation using the superscript convention from Chapter 8: $\mathcal{R}^{\text{deon}} = \mathcal{R}_{|\text{deon}(\mathcal{R})}$. Similarly for $\mathcal{R}^{\text{epist}}$ and $\mathcal{R}^{\text{circ}}$ (circumstantial). The lexical entry for a deontic modal like *have to* contains this free-but-constrained variable:

(51) $\textit{have to} \rightsquigarrow \lambda p\,.\,\forall w'[\mathcal{R}^{\text{deon}}_{w_0}(w') \rightarrow p(w')]$

The assignment function g supplies the value of $\mathcal{R}$, picking out whichever deontic accessibility relation is relevant in context.

Assuming *necessarily* is polysemous, with different lexical entries for each flavor, the deontic sense is:

(52) $\textit{necessarily}_{\text{DEON}} \rightsquigarrow \lambda p_{st}\,.\,\forall w'[\mathcal{R}^{\text{deon}}_{w_0}(w') \rightarrow p(w')]$

Epistemic and circumstantial necessity work analogously:

(53) $\textit{necessarily}_{\text{EPIST}} \rightsquigarrow \lambda p_{st}\,.\,\forall w'[\mathcal{R}^{\text{epist}}_{w_0}(w') \rightarrow p(w')]$

(54) $\textit{necessarily}_{\text{CIRC}} \rightsquigarrow \lambda p_{st}\,.\,\forall w'[\mathcal{R}^{\text{circ}}_{w_0}(w') \rightarrow p(w')]$

The corresponding senses of *possibly* are analogous, using existential instead of universal quantification:

(55) $\textit{possibly}_{\text{CIRC}} \rightsquigarrow \lambda p_{st}\,.\,\exists w'[\mathcal{R}^{\text{circ}}_{w_0}(w') \wedge p(w')]$

12.5.2 Propositional attitudes

Like modals, so-called 'propositional attitude' verbs like *believe* and *want* expect intensional inputs of type $\langle s,t\rangle$. For propositional attitudes, the accessibility relation is relativized to an individual agent. We introduce two further flavor predicates:

- dox, type $\langle\langle s,\langle s,t\rangle\rangle,t\rangle$: true of doxastic accessibility relations (those compatible with an agent's beliefs; from Greek *doxa*, 'belief')
- boul, type $\langle\langle s,\langle s,t\rangle\rangle,t\rangle$: true of bouletic accessibility relations (those compatible with an agent's desires; from Greek *boulē*, 'will, desire')

The agent-specific doxastic relation for individual x is $\mathcal{R}^{\mathsf{dox}}_x$, which by the superscript convention expands to $\mathcal{R}(x)_{|\mathsf{dox}(\mathcal{R}(x))}$, where the full free variable $\mathcal{R}$ has type $\langle e,\langle s,\langle s,t\rangle\rangle\rangle$ and $\mathcal{R}(x)$ (the x-slice) has type $\langle s,\langle s,t\rangle\rangle$, to which dox applies (cf. Hintikka 1969). If a world w' belongs to $\mathcal{R}^{\mathsf{dox}}_x(w_0)$, we say that w' is DOXASTICALLY ACCESSIBLE to x at w_0. To believe a proposition p, then, is for p to hold at all doxastically accessible worlds:

(56) $\quad believes \rightsquigarrow \lambda p_{st}\lambda x.\,\forall w'[\mathcal{R}^{\mathsf{dox}}_{x,w_0}(w') \rightarrow p(w')]$

We abbreviate $\forall w'[\mathcal{R}^{\mathsf{dox}}_{\alpha,u}(w') \rightarrow p(w')]$ as $\mathsf{bel}_u(\alpha,p)$.

Similarly, $\mathcal{R}^{\mathsf{boul}}_x = \mathcal{R}(x)_{|\mathsf{boul}(\mathcal{R}(x))}$ is the bouletic (desire) accessibility relation for agent x. To want a proposition p is for p to hold at all of x's desire worlds:

(57) $\quad wants \rightsquigarrow \lambda p_{st}\lambda x.\,\forall w'[\mathcal{R}^{\mathsf{boul}}_{x,w_0}(w') \rightarrow p(w')]$

We abbreviate $\forall w'[\mathcal{R}^{\mathsf{boul}}_{\alpha,u}(w') \rightarrow p(w')]$ as $\mathsf{want}_u(\alpha,p)$.

Exercise 6. *The following exercise is adapted from von Fintel & Heim (2011), p. 21. You may find the surrounding discussion in that chapter helpful in doing this exercise.*

Consider the following two alternative ways of defining the semantics of *believes*:

(i) *believes* $\rightsquigarrow \lambda p_{st} \lambda x . \forall w' [\mathcal{R}^{\mathsf{dox}}_{x,w_0}(w') \leftrightarrow p(w')]$

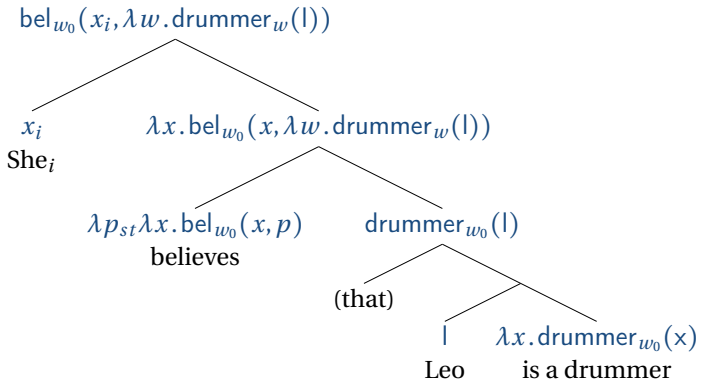
(ii) *believes* $\rightsquigarrow \lambda p_{st} \lambda x . \exists w' [\mathcal{R}^{\mathsf{dox}}_{x,w_0}(w') \wedge p(w')]$

Both are very wrong. Explain why these do not adequately capture the meaning of *believe*.

With these rules, we derive the representation in (59) for (58).

(58) She believes that Leo is a drummer.

(59)



Note that the world variable w_0 was re-lettered after it was bound by the λ-operator, so instead of $\lambda w_0 . \mathsf{drummer}_{w_0}(\mathsf{l})$, we have written $\lambda w . \mathsf{drummer}_w(\mathsf{l})$.

12.6 De dicto vs. De re

12.6.1 The distinction

Recall (38b), repeated here:

(60) Andrea wants a sloop.

Although this sentence has a reading on which it does not commit the speaker to the existence of sloops, there is another reading, too. It could also be interpreted to mean that there is a particular sloop that Andrea wants (as in *Andrea wants a sloop, namely mine*). The two readings involved here are called DE RE ('of the object') and DE DICTO ('of the word').[3] On the *de re* reading, Andrea has a desire for a particular sloop: Regarding *that* sloop, she wants it. The *de dicto* reading is the one on which she merely seeks relief from slooplessness. In the latter case, the desire is not about a particular object, rather it is about whether Andrea ends up having a sloop. In this sense, Andrea's desire is for the false proposition *that Andrea has a sloop* to become true.[4]

Quine (1956) also illustrated the *de dicto* / *de re* ambiguity with

[3] A similar distinction may have been anticipated by Aristotle and is discussed by the medieval logician Abelard. The terms themselves appear for the first time about one century later in the writings of Saint Thomas Aquinas, the philosopher and theologian.

[4] The *de re*/*de dicto* distinction is related to the distinction between specific and nonspecific indefinites. In many languages, indefinites can be marked for specificity using what is known as Differential Object Marking (DOM). For example, in Spanish, the object of verbs like *buscar* "look for" is optionally marked by the preposition *a* to indicate specificity:

(i) a. Juan busca a un profesor.
b. Juan busca un profesor.

Sentence (ia) expresses that there is a specific teacher Juan is looking for (so *un professor* is understood *de re*), while (ib) expresses that Juan is not looking for any teacher in particular but merely wants the proposition *that he finds a teacher* to become true (so *un professor* is understood *de dicto*).

examples like the following:

(61) a. Ralph believes that someone is a spy.
b. Ralph believes that the man in the brown hat is a spy.

Consider first example (61a). On the *de re* reading, Ralph has a belief about a particular individual: There is someone about whom Ralph believes that this person is a spy. On the *de dicto* reading, Ralph has no particular individual in mind; he just believes that there are spies. The belief is not about a particular individual, rather it's about the proposition that there is a spy. Ralph believes that the proposition is true. Neither interpretation entails that there are, in fact, any spies; on either reading, Ralph's belief could simply be mistaken. What the *de re* reading additionally entails is that there is some particular individual about whom Ralph holds the belief — not that this individual is in fact a spy. The *de dicto* reading does not require even that. For example (61b), the difference between the two readings comes down to whether it is Ralph (in the *de dicto* reading) or the speaker (in the *de re* reading) who would describe the person in question as wearing a brown hat.

Not only propositional attitude verbs but modals too give rise to *de re* / *de dicto* ambiguities. To understand the point of the following example, you have to know that the first man in space was the Russian Yuri Gagarin.

(62) The first man in space might have been an American. (Evans, 1977)

This sentence is ambiguous. When the definite description *the first man in space* is read *de dicto* with respect to the modal *might*, the resulting reading can be paraphrased as *It might have been the case that an American is the first man in space*; that is, the Americans could have won this episode of the space race. When it is read *de re*, the resulting reading can be paraphrased as *Concerning the first man who actually reached space, he might have been*

an American; that is, Gagarin could have been a U.S. citizen.

12.6.2 Towards an analysis of *de dicto* vs. *de re*

To develop a treatment of the *de dicto* vs. *de re* ambiguity, let us focus on the following example.

(63) Mariel wants to dance with a drummer.

On one interpretation, there is a specific drummer who Mariel wants to dance with. Mariel may not even *know* that the person in question is a drummer; her desire is just to dance with that person. This is the *de re* interpretation, as her desire is *de re* with respect to a particular individual. On the *de dicto* interpretation, there is no specific drummer that Mariel want to dance with; her desire is for the proposition that Mariel dances with a drummer to become true. This desire would be satisfied in any world where there is some drummer, any drummer, that she is dancing with.

Exercise 7. Consider the following case from Anderson (2014):

> In 1869, an English court considered the case of Whiteley v. Chappell, in which a man who had voted in the name of his deceased neighbor was prosecuted for having fraudulently impersonated a "person entitled to vote." The court acquitted him, albeit reluctantly. There had been voter fraud by impersonation, certainly. But the court fixated on the object of the impersonation and concluded that because a dead person could not vote, the defendant had not impersonated a "person entitled to vote." The court attributed the mismatch between this result and the evident purpose of the statute to an oversight of the drafters: "The legislature has not used words wide

> enough to make the personation of a dead man an offence."

How would you characterize the *de re* and *de dicto* interpretations, respectively, in this case? Which interpretation does the court appear to have taken? Is there an interpretation on which the man is guilty? Explain why or why not.

The *de re* reading of this example can be paraphrased: 'There is an entity x such that x is a drummer and Mariel wants to dance with x.' The *de dicto* reading can be paraphrased, 'What Mariel wants is for it to be true that there is a drummer that she dances with.' Under what circumstances, exactly, would each of these readings be considered true?

In constructing a semantics for desire claims, it is useful to keep in mind that desires are contingent rather than necessary states of affairs. It is perfectly reasonable to imagine two distinct possible worlds w and w', which differ with respect to what Mariel's desires are in them. In other words, claims about desire can be true in one world and false in another.

Let us assume that the meaning of the verb *want* can be construed in terms of a ternary relation between:

- an individual x (Mariel in this example)
- a world w (the world of evaluation), and
- a set of worlds, containing all and only those worlds in which all the desires that the individual has at w are fulfilled (the 'desire worlds').

As desire is a *bouletic*-flavored genre of modality, those worlds w' can be called the BOULETICALLY ACCESSIBLE POSSIBILITIES (for x in w), or just x's DESIRE WORLDS (in w) for short. If Mariel wants a proposition ϕ, then ϕ holds in all of her desire worlds. On the

analysis we will develop, the two readings of (63) will be analyzed as follows:

- The *de dicto* reading is true in a given world of evaluation w just in case, in every desire world w' of Mariel's in w, she dances *in the desire world* w' with someone who is a drummer *in the desire world* w'.
- The *de re* reading is true in a given world of evaluation w just in case there is some individual x who is a drummer *in the world of evaluation* w and in every desire world w' of Mariel's in w, she dances with x.

Recall that our representation language Ty2 is interpreted relative to a model that specifies a domain of individuals, an interpretation function, and a set of possible worlds. For purposes of illustration, let us consider a model like this where in some of the worlds in W, Mariel dances with a drummer. Let us assume that there are only four worlds, which we will call World A, World B, World C, and World D. Assume that in World B and World C, Mariel dances only with Ruth. In World D, she dances only with Leo. In World A, she dances with nobody. This information, along with who is a drummer in each world, is summarized in Table 12.1. Notice that in World C, Mariel dances only with Ruth, and since Ruth is not a drummer in that world, it is not true in World C that Mariel dances with a drummer. In World D, Mariel dances only with Leo, and since Leo is a drummer in World D, it's true in World D that Mariel dances with a drummer.

Recall that Mariel's desires may well vary from world to world just as other things do. Suppose that in World D, she doesn't want to dance with anybody, even though she happens to be dancing with Leo. In other words, in World D, the only world among her desire worlds (the worlds that satisfy her desires) is World A. Mariel's desire worlds in each world are likewise specified as in Table 12.1. Building on our assumption that x *wants* ϕ in w is true if and only if ϕ holds in all of the individual's desire worlds, we have all of

	World A	World B	World C	World D
drummers	Ruth	Ruth	Leo	Leo
Mariel dances with	nobody	Ruth	Ruth	Leo
Mariel desires	{B, D}	{B, C}	{B, C}	{A}
(63) *de dicto*	T	F	F	F
(63) *de re*	F	T	F	F

Table 12.1: Above midline: Partial specification of a model for Ty2. Below midline: truth status of two readings of (63) *Mariel wants to dance with a drummer* in each world. *Mariel dances with a drummer* is true in World B and World D.

the information we need in order to evaluate the truth or falsity of (63) (*Mariel wants to dance with a drummer*) on various readings at various worlds.

In World D, the sentence is false on any reading. In World D, Mariel just doesn't want to dance with anybody, drummer or not.

Notice that in World A, Mariel's desire worlds are World B (where she dances with Ruth, a drummer in that world) and World D (where she dances with Leo, a drummer in that world). Then, in every one of her desire worlds, she dances with somebody who is a drummer in that world. That means that, relative to World A, the *de dicto* reading of (63) is true.

In World B, Mariel's desire worlds are World B and World C. In both of these worlds, she dances with Ruth, who happens to be a drummer in World B, though not in World C. In World B, then, Mariel wants *de re* to dance with someone, namely Ruth; and Ruth is a drummer. So the *de re* reading of (63) is true in World B. This is so even though Ruth is not a drummer in World C — but the *de dicto* reading of (63) is false in World B for that reason.

Finally, in World C, Mariel's desire worlds are again World B and World C. She just wants to dance with Ruth. But Ruth is not

a drummer in World C. So although in World C Mariel does want *de re* to dance with *someone* (namely Ruth), it is not the case that she wants *de re* to dance with a *drummer*. So the *de re* reading of *Mariel wants to dance with a drummer* is false in World C, and is its *de dicto* reading.

12.6.3 An analysis of *de dicto* vs. *de re*

Let us now consider how to compositionally derive the two readings of *She wants to dance with a drummer*. Since *want* takes a non-finite complement in this example, we will need to establish some assumptions about the syntax of non-finite complements. There are a number of choices one could make here, but we will simply assume that there is a silent pronoun PRO_i in the subject position coindexed with the subject of the sentence.[5] This gives:

(64) She_i wants [PRO_i (to) dance with a drummer]

Then *want* combines in effect with a full clause. Whether we get a *de dicto* or *de re* reading depends on the landing site of *a drummer* when it undergoes QR. The *de dicto* reading can be derived from an LF structure where *a drummer* remains under the scope of *wants*:

(65) She_1 wants [[a drummer] λ_2 PRO_1 (to) dance with t_2]

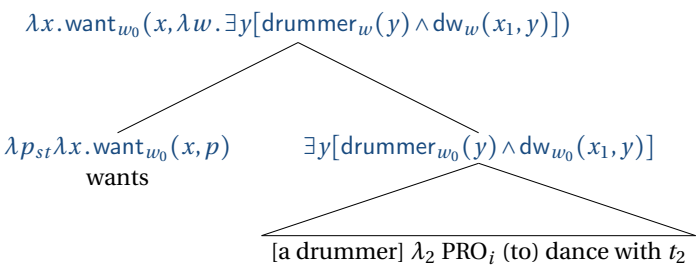
The *de re* reading involves higher scope for *a drummer* at LF:

(66) [a drummer] λ_2 she_1 wants [PRO_1 (to) dance with t_2]

Our composition rules deliver the following derivation for the LF in (65) (dw is short for 'dance with'):

[5] The example we are discussing is a case of 'control', as opposed to 'raising'; 'subject control' in particular, since it is the subject of the sentence that 'controls' the interpretation of the embedded subject. See Carnie (2013) for an introduction to raising and control.

(67)



The translation for the full sentence on the *de dicto* reading is as follows:

(68) $\mathsf{want}_{w_0}(x_1, \lambda w.\exists y[\mathsf{drummer}_w(y) \wedge \mathsf{dw}_w(x_1, y)])$

The derived truth conditions can be paraphrased as: 'In the world of evaluation (= w_0), she (= x_1) stands in the 'want' relation to the proposition that holds in any world w such that there is a drummer in w that she (= x_1) dances with in w.' That is to say, she wants the proposition denoted by the sentence "She dances with a drummer" to be true.

The truth of the statement in (68) depends on a model and an assignment function. Whether or not it is true relative to model M and assignment function g depends in particular on what g assigns to both x_1 and w_0. If g assigns x_1 to Mariel and w_0 to World A, and according to M, World A is as described in Table 12.1, then this formula will be true relative to M and g.

Exercise 8. Using an appropriate LF, give a compositional derivation for the *de re* reading of *She wants to dance with a drummer*, and give an example of an assignment function relative to which it would be true given the model described in Table 12.1.

12.7 The Ty2 formal system

Ty2 (Gallin, 1975) is a version of the simply typed lambda calculus that adds a primitive type s for possible worlds, enabling expressions of the language to denote, quantify over, and abstract over worlds directly. It is the representation language used throughout this chapter.

12.7.1 Syntax of Ty2

Types

The TYPES of Ty2 are defined as follows:

- e, t, and s are primitive types, corresponding to individuals, truth values, and possible worlds respectively.
- If σ and τ are types, then $\langle \sigma, \tau \rangle$ is a type.
- Nothing else is a type.

A FORMULA is an expression of type t. We write w_0 for the designated evaluation-world variable (a variable of type s).

Basic expressions

For every type τ, Ty2 provides a set of NON-LOGICAL CONSTANTS Con_τ, with one constant $c_{n,\tau}$ for each natural number n, and a set of VARIABLES Var_τ, with one variable $v_{n,\tau}$ for each natural number n.

Expression-forming rules

All remaining expressions are formed by the following rules:

1. **Application**
   For any types σ and τ, if α is an expression of type $\langle \sigma, \tau \rangle$ and

β is an expression of type σ, then $[\alpha(\beta)]$ is an expression of type τ.

When β is of type s, we write α_β as shorthand for $[\alpha(\beta)]$.

2. **Identity**
If α and β are expressions of the same type, then $\alpha = \beta$ is a formula.

3. **Negation**
If ϕ is a formula, then so is $\neg\phi$.

4. **Binary Connectives**
If ϕ and ψ are formulas, then so are $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \rightarrow \psi]$, and $[\phi \leftrightarrow \psi]$.

5. **Quantification**
If ϕ is a formula and u is a variable of any type, then $[\forall u.\phi]$ and $[\exists u.\phi]$ are formulas.

6. **Lambda Abstraction**
If α is an expression of type τ and u is a variable of type σ, then $[\lambda u.\alpha]$ is an expression of type $\langle\sigma,\tau\rangle$.

12.7.2 Semantics of Ty2

Model

A TY2 MODEL is a triple

$$M = \langle (D_\tau)_\tau,\ I,\ W \rangle$$

where $(D_\tau)_\tau$ is a STANDARD FRAME: an indexed family of domains, one per type, defined by the following primitive-type assignments together with the rule that $D_{\langle\sigma,\tau\rangle}$ is the set of all functions from D_σ to D_τ:

- $D_e = D$ (a non-empty set of individuals)

- $D_s = W$ (a non-empty set of possible worlds)
- $D_t = \{\mathsf{T}, \mathsf{F}\}$ (the truth values)

Note that D is a single constant domain across all worlds. The idea that an individual might exist in one world but not another can be expressed using a world-dependent existence predicate of type $\langle s, \langle e, t\rangle\rangle$. This commits us to a POSSIBILIST ontology rather than an ACTUALIST one.[6]

The interpretation function I assigns to each non-logical constant of type τ a value in D_τ.

Denotation clauses

An ASSIGNMENT is a function assigning to each variable of type τ a value in D_τ. Semantic evaluation is relative to a model M and an assignment g; we write $[\![\alpha]\!]^{M,g}$.

1. **Basic Expressions**
   (a) If α is a non-logical constant: $[\![\alpha]\!]^{M,g} = I(\alpha)$.
   (b) If α is a variable: $[\![\alpha]\!]^{M,g} = g(\alpha)$.
2. **Application**
   $[\![[\alpha(\beta)]]\!]^{M,g} = [\![\alpha]\!]^{M,g}\big([\![\beta]\!]^{M,g}\big)$.
3. **Identity**
   $[\![\alpha = \beta]\!]^{M,g} = \mathsf{T}$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$.
4. **Negation**
   $[\![\neg\phi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{F}$.
5. **Binary Connectives**
   (a) $[\![\phi \wedge \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{T}$ and $[\![\psi]\!]^{M,g} = \mathsf{T}$.
   (b) $[\![\phi \vee \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{T}$ or $[\![\psi]\!]^{M,g} = \mathsf{T}$.

[6] For more on possibilism vs. actualism, see Menzel (2018).

(c) $[\![\phi \rightarrow \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = \mathsf{F}$ or $[\![\psi]\!]^{M,g} = \mathsf{T}$.

(d) $[\![\phi \leftrightarrow \psi]\!]^{M,g} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g} = [\![\psi]\!]^{M,g}$.

6. **Quantification**

   (a) $[\![\forall u.\phi]\!]^{M,g} = \mathsf{T}$ iff for all $o \in D_\tau$: $[\![\phi]\!]^{M,g[u \mapsto o]} = \mathsf{T}$ (where τ is the type of u).

   (b) $[\![\exists u.\phi]\!]^{M,g} = \mathsf{T}$ iff there exists $o \in D_\tau$ such that $[\![\phi]\!]^{M,g[u \mapsto o]} = \mathsf{T}$.

7. **Lambda Abstraction**
   $[\![\lambda u.\alpha]\!]^{M,g}$ is that function f from D_σ to D_τ such that for all $o \in D_\sigma$: $f(o) = [\![\alpha]\!]^{M,g[u \mapsto o]}$ (where u has type σ and α has type τ).

12.8 Summary

This chapter has developed a compositional intensional semantics, using Ty2 as a representation language. We began with two dimensions along which modal expressions vary: *flavor* (deontic, epistemic, circumstantial, bouletic, and so on) and *strength* or *force* (necessity versus possibility). Strong and weak modals stand in the same dual relationship to one another as the universal and existential quantifiers do, regardless of their flavor. We also distinguished necessary from contingent truth, and observed that some environments are *opaque*: substituting one coextensional expression for another can change the truth value of the whole. Such failures of substitutability motivate a semantics on which an expression has not just an *extension* but also an *intension*, a function from possible worlds to extensions of the appropriate type.

To model this formally, we introduced modal logic, in which the operators $\Box$ and $\Diamond$ quantify over the worlds made available by an *accessibility relation* in a *Kripke model*. We then moved to Ty2, which dispenses with these operators in favor of explicit quantification over possible worlds, made possible by a primitive type

s for worlds. Ty2 is more expressive than modal logic in this respect, since a single formula can refer to more than one world at a time. On the compositional side, we saw that certain heads select for the intension of their complement rather than its extension; building on the tradition of Montague's *hat operator*, we adopted the rule of Intensional Function Application to handle such cases.

With this machinery in place, modal flavor could be treated as a matter of which accessibility relation is in play, and propositional attitude verbs could be analyzed in the same terms, with *believe* quantifying over an agent's doxastically accessible worlds and *want* over their desire worlds. We then analyzed the *de dicto* / *de re* ambiguity as a matter of scope, according to whether the indefinite is interpreted inside or outside the scope of the attitude verb or modal. For a book-length presentation of intensional semantics, we recommend von Fintel & Heim (2011).

13 | Synthesis

Chapter 12 introduced Ty2 and used it to model intensional semantics: possible worlds, necessity, possibility, and de dicto/de re ambiguity. This chapter synthesizes the developments of the preceding chapters into a single, integrated framework. Section 13.1 combines the tense–aspect system of Chapter 11 with the intensional system of Chapter 12. Section 13.2 addresses the treatment of propositional attitude verbs in an event-semantic setting, building on recent work by Kratzer (2016), Moulton (2009), and Bondarenko (2022). The full formal system Ty2^{+}, which integrates all the primitive types needed across the book, is defined in Section 13.3. Finally, Section 13.4 presents a unified fragment that draws together all of the semantic tools developed in the book.

13.1 Tense and modality

The modal entries developed in this chapter treat modals as quantifiers over sets of worlds, ignoring time entirely. But modal sentences are systematically sensitive to tense, as the two readings of the following example show (Condoravdi, 2002; Rullmann & Matthewson, 2017):

(1) There **might have** been ice cream in the freezer.

On one reading, the sentence expresses present epistemic uncertainty about a past state: right now, for all the speaker knows,

there was ice cream. On the other reading, it expresses a past circumstantial possibility: at some time in the past, circumstances were open in a way that permitted ice cream being there. These two readings differ in modal flavor (epistemic vs. circumstantial) and in two temporal parameters that Rullmann & Matthewson (2017) call TEMPORAL PERSPECTIVE and TEMPORAL ORIENTATION.

The TEMPORAL PERSPECTIVE (TP) is the time at which the modal base is evaluated — the vantage point from which possibilities are assessed. In the epistemic reading of (1), the TP is the present: the speaker's current epistemic state is the relevant one. In the circumstantial reading, the TP is some past time: circumstances at that past moment are what matters.

The TEMPORAL ORIENTATION (TO) is the relation between the TP and the time of the eventuality described by the modal's prejacent. In the epistemic reading, the TO is past: the potential state of affairs (ice cream in the freezer) precedes the TP. In the circumstantial reading, the TO is future: the open possibility lies ahead of the past TP.

Rullmann and Matthewson's key architectural proposal is that TP and TO are determined by distinct grammatical mechanisms. The TP is contributed by a temporal operator *above* the modal — typically tense — and the TO is contributed by an aspectual operator *below* the modal:

(2) tense > modal > ordering aspect > inclusion aspect > vP

This hierarchy slots naturally into the two-layer aspect system of Chapter 11.

Extended modal entries. To implement this, modals must take a *predicate of times* as their argument rather than a proposition. Combining the tense–aspect system of Chapter 11 with the intensional framework of this chapter, a vP denotes a function from times to propositions, type $\langle i, \langle s, t\rangle\rangle$. A modal takes such a function and returns one of the same type, so modals have type $\langle\langle i, \langle s, t\rangle\rangle, \langle i, \langle s, t\rangle\rangle\rangle$.

The accessibility relation must be parameterized by both a world and a time, written $\mathcal{R}_{w_0,t}$, capturing the idea that what is epistemically possible (or deontically required) can differ across times. The entry for *have to* on its epistemic reading becomes:

(3) $\textit{have to}_{\text{EPIST}} \rightsquigarrow \lambda P_{\langle i,st\rangle}.\,\lambda t.\,\lambda w_0.\,\forall w'[\mathcal{R}^{\text{epist}}_{w_0,t}(w') \rightarrow P(t)(w')]$
$\langle\langle i,st\rangle,\langle i,st\rangle\rangle$

The reference time t comes from the tense head T′ above the modal. For a past-tense sentence like *This **had to** be a mistake!*, T′ denotes t_n^{past} (type i, from Chapter 11). Applying the modal entry to the vP predicate P and then to T′ yields:

(4) $\forall w'[\mathcal{R}^{\text{epist}}_{w_0,t_n^{\text{past}}}(w') \rightarrow P(t_n^{\text{past}})(w')]$

This says: at the past reference time t_n^{past}, every epistemically accessible world is one in which P holds at t_n^{past}. The result is a proposition (type $\langle s,t\rangle$) asserting past epistemic necessity.

A cross-linguistic preview. R&M's architecture predicts that TP and TO should be independent of modal flavor across languages, with both parameters derived compositionally from independently motivated tense and aspect operators. Data from three languages support this picture.

Dutch provides the clearest illustration: Dutch modal verbs inflect for tense just like main verbs. The modals *moeten* ('must') and *kunnen* ('can/might') have simple past forms *moest* and *kon*. In the right discourse context, both can have epistemic readings with past TP. Since the tense morphology is unambiguous, the past-TP epistemic reading is entirely transparent compositionally.

Gitksan (Interior Tsimshianic, spoken in northwestern British Columbia) offers a different kind of evidence. It has no overt past/present tense distinction but overtly marks future TO with the prospective aspect marker *dim*. Crucially, it lexically distinguishes epistemic modals (=*imaa*) from circumstantial ones (*da'akhlxw*, *sgi*). Be-

cause the flavor is morphologically unambiguous, the data show cleanly that any flavor can receive a past-TP reading in context — and that circumstantial modals require *dim* for future TO, while epistemic modals do not.

St'át'imcets (Lilooet Salish, also in British Columbia) patterns similarly: no overt past/present tense distinction, overt prospective aspect (*cuz'*, *kelh*), and lexically distinct epistemic (*k'a*) vs. circumstantial (*ka*) modals. Both flavors freely allow past TP in context; future TO requires the prospective marker.

In all three languages, the TP/TO architecture falls out from independently needed tense and aspect operators. English is more idiosyncratic — modal auxiliaries do not straightforwardly inflect for tense — but the same underlying parameters are operative. See Rullmann & Matthewson (2017) for detailed formal analyses and cross-linguistic comparison.

13.2 Attitude verbs in event semantics

Attitude verbs like *believe* make use of an accessibility relation that is anchored to the belief-holder, expressed as the subject of the verb. In Chapter 12, *believe* was treated as a function that takes a proposition p and an individual x directly. This makes it easy to express the constraint that the content of the belief holds in all of the doxastically accessible worlds for the belief-holder. When we move to a Neo-Davidsonian event-semantic framework, the compositional link between the belief-holder and the content of the belief is broken. So the question arises how to derive compositionally that it is the experiencer of the belief whose doxastic alternatives are constrained by the content of the belief.

Luckily, there is a different, and arguably better-motivated, analysis that naturally leads to a solution to this compositional problem. This analysis builds on a striking parallel in the nominal domain (Kratzer, 2016; Moulton, 2009; Bondarenko, 2022).

The nominal motivation. Nouns like *belief*, *fact*, *claim*, and *rumor* denote CONTENT-BEARING INDIVIDUALS — entities with propositional content. The *that*-clause in a nominal like (5) specifies what that content is.

(5) the belief that Leo is a drummer

This *that*-clause functions as a MODIFIER, not as an argument. The contrast with genuine nominal arguments is telling: the object of *destruction* in (6a) can pronominalize with *of*, while the *that*-clause in (5) cannot.

(6) a. the destruction of Rome →
the destruction of *it*
b. the belief that Leo is a drummer →
*the belief of *it*

The *that*-clause in (5) is better analyzed as an APPOSITIVE that specifies the propositional content of the entity denoted by *belief*. Such entities — call them CONTENT OBJECTS — are individuals that bear a proposition as their content.

The content function. We add to the model a function cont of type $\langle e, st\rangle$, mapping content objects to their propositional content (a set of possible worlds). The *that*-clause, treated as a CONT-CP, is then a predicate of content objects whose content equals the embedded proposition:

(7) *that* $S \rightsquigarrow \lambda b.\mathsf{cont}(b) = \lambda w_0.\phi$, type $\langle e, t\rangle$
where $S \rightsquigarrow \phi$

So (5) denotes the unique content object that is a belief with the right content: $\iota b[\mathsf{belief}(b) \wedge \mathsf{cont}(b) = \lambda w.\mathsf{drummer}_w(\mathsf{l})]$.

Attitude verbs in event semantics. When we combine this picture with the event-semantic framework of Chapter 10, the redun-

dancy noted there is resolved cleanly. A believing event has *two* participant roles:

- an EXP participant — the believer, introduced by the EXP θ-role head via IND+ as usual
- a THEME participant — the content object, a belief-individual whose content is the embedded proposition

The verb itself is simply a predicate of believing events:

(8) *believes* $\rightsquigarrow \lambda V.\exists e.\mathsf{believe}(e) \wedge V(e)$, type $\langle\langle v,t\rangle,t\rangle$

The THEME θ-role head shifts via PRED+ with the Cont-CP as its restricting predicate. For the sentence *Mary believes that Leo is a drummer*, where $p = \lambda w.\mathsf{drummer}_w(\mathsf{l})$, the result after combining the Cont-CP with the θ-head and the verb (and before EQ applies) is:

(9) $\lambda V.\exists e.[\mathsf{believe}(e) \wedge \mathsf{cont}(\mathsf{theme}(e)) = p \wedge V(e)]$

The EXP θ-role head then introduces Mary via IND+ in the standard way.

Where the doxastic relation goes. The doxastic accessibility relation is no longer part of the verb's lexical entry. It becomes a MODEL-THEORETIC CONSTRAINT on what events count as believing events: for any believing event e in any model, all worlds w' doxastically accessible for the experiencer $\mathsf{exp}(e)$ at the world and time of the event $\langle\chi(e),\tau(e)\rangle$ satisfy the content of the belief, written $\mathsf{cont}(\mathsf{theme}(e))$.

13.3 The Ty2$^+$ formal system

Ty2$^+$ is a version of the simply typed lambda calculus designed for natural language semantics. It extends Ty2 (Gallin, 1975)—which adds explicit reference to and quantification over possible worlds

to L_λ—with primitive types for times, events, and natural numbers.

13.3.1 Syntax of Ty2$^+$

Types

The TYPES of Ty2$^+$ are defined as follows:

- The following are primitive types: e (individuals), t (truth values), s (possible worlds), i (time intervals), v (events), n (natural numbers).
- If σ and τ are types, then $\langle \sigma, \tau \rangle$ is a type.
- Nothing else is a type.

A FORMULA is an expression of type t.

Basic expressions

For every type τ, Ty2$^+$ provides a set of NON-LOGICAL CONSTANTS Con_τ, with one constant $c_{n,\tau}$ for each natural number n, and a set of VARIABLES Var_τ, with one variable $v_{n,\tau}$ for each natural number n.

We write w_0 for the designated evaluation-world variable (a variable of type s).

Expression-forming rules

All remaining expressions are formed by the following rules:

1. **Application**
   For any types σ and τ, if α is an expression of type $\langle \sigma, \tau \rangle$ and β is an expression of type σ, then $[\alpha(\beta)]$ is an expression of type τ.

   When β is of type s, we write α_β as shorthand for $\alpha(\beta)$.

2. **Identity**
   If α and β are expressions of the same type, then $\alpha = \beta$ is a formula.

3. **Negation**
   If ϕ is a formula, then so is $\neg\phi$.

4. **Binary Connectives**
   If ϕ and ψ are formulas, then so are $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \rightarrow \psi]$, and $[\phi \leftrightarrow \psi]$.

5. **Quantification**
   If ϕ is a formula and u is a variable of any type, then $[\forall u.\phi]$ and $[\exists u.\phi]$ are formulas.

6. **Lambda Abstraction**
   If α is an expression of type τ and u is a variable of type σ, then $[\lambda u.\alpha]$ is an expression of type $\langle\sigma,\tau\rangle$.

7. **Temporal Ordering**
   If α and β are expressions of type i, then $\alpha < \beta$ and $\alpha \subseteq \beta$ are expressions of type t.

8. **Mereological Expressions**
   If α and β are expressions of type e and π is an expression of type $\langle e,t\rangle$:

   - $\alpha \sqsubseteq \beta$ is an expression of type t (part-of)
   - $\alpha \oplus \beta$ is an expression of type e (binary sum)
   - $\bigoplus\pi$ is an expression of type e (generalized sum)
   - ${}^*\pi$ is an expression of type $\langle e,t\rangle$ (algebraic closure)

9. **Presupposition**
   If ϕ is a formula, then $\partial(\phi)$ is a formula (the presupposition operator from L_∂, Chapter 8; type $\langle t,t\rangle$).

Abbreviations:

- $\alpha \leq \beta$ for $\alpha < \beta \vee \alpha = \beta$ (temporal precedence-or-simultaneity)
- $\alpha \sqsubset \beta$ for $\alpha \sqsubseteq \beta \wedge \alpha \neq \beta$ (proper part)
- $\mathsf{atom}(\alpha)$ for $\neg\exists y[y \sqsubset \alpha]$ (atomic individual)

13.3.2 Semantics of Ty2$^+$

Model

A TY2$^+$ MODEL is a tuple

$$M = \langle (D_\tau)_\tau,\ I,\ T,\ <,\ \subseteq,\ \sqsubseteq,\ E,\ W,\ C \rangle$$

where $(D_\tau)_\tau$ is a STANDARD FRAME: an indexed family of domains, one per type, defined by the following primitive-type assignments together with the rule that $D_{\langle\sigma,\tau\rangle}$ is the set of all functions from D_σ to D_τ:

- $D_e = D$ (a non-empty set of individuals)
- $D_i = T$ (a non-empty set of time intervals)
- $D_v = E$ (a non-empty set of events)
- $D_s = W$ (a non-empty set of possible worlds)
- $D_n = \mathbb{N}$ (the natural numbers)
- $D_t = \{\mathsf{T}, \mathsf{F}\}$ (the truth values)

The remaining components of M are:

- An INTERPRETATION FUNCTION I that assigns to each non-logical constant of type τ a value in D_τ.
- A strict linear order $<$ on T (temporal precedence) and a containment relation $\subseteq$ on T (i.e. $t_1 \subseteq t_2$ iff every moment in t_1 is also in t_2).

- A partial order $\sqsubseteq$ on D (the individual-part relation), which uniquely determines the binary sum $\oplus$, generalized sum $\bigoplus$, and algebraic closure * on D.
- A non-empty set C of CONTEXTS OF UTTERANCE. Each $c \in C$ is a tuple

$$c = \langle \mathsf{sp}(c), \mathsf{ad}(c), \mathsf{time}(c), \mathsf{loc}(c), \mathsf{world}(c) \rangle$$

  where $\mathsf{sp}(c) \in D$ is the speaker, $\mathsf{ad}(c) \in D$ is the addressee, $\mathsf{time}(c) \in T$ is the time of utterance, $\mathsf{loc}(c) \in D$ is the location, and $\mathsf{world}(c) \in W$ is the world of utterance.

Two non-logical constants of $\mathrm{Ty2}^+$ have distinguished roles as structural functions:

- $\tau \in \mathrm{Con}_{\langle v,i \rangle}$: the RUNTIME FUNCTION, mapping each event to the time interval during which it occurs
- $\chi \in \mathrm{Con}_{\langle v,s \rangle}$: the WORLD FUNCTION, mapping each event to the possible world in which it occurs

Denotation clauses

An ASSIGNMENT is a function assigning to each variable of type τ a value in D_τ. Semantic evaluation is relative to a model M, an assignment g, and a context c; we write $[\![\alpha]\!]^{M,g,c}$.

1. **Basic Expressions**
   (a) If α is a non-logical constant: $[\![\alpha]\!]^{M,g,c} = I(\alpha)$.
   (b) If α is a variable: $[\![\alpha]\!]^{M,g,c} = g(\alpha)$.
2. **Application**
   $[\![[\alpha(\beta)]]\!]^{M,g,c} = [\![\alpha]\!]^{M,g,c}\big([\![\beta]\!]^{M,g,c}\big)$.
3. **Identity**
   $[\![\alpha = \beta]\!]^{M,g,c} = \top$ iff $[\![\alpha]\!]^{M,g,c} = [\![\beta]\!]^{M,g,c}$.

4. **Negation**
   $[\![\neg\phi]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g,c} = \mathsf{F}$.

5. **Binary Connectives**

   (a) $[\![\phi \wedge \psi]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g,c} = \mathsf{T}$ and $[\![\psi]\!]^{M,g,c} = \mathsf{T}$.

   (b) $[\![\phi \vee \psi]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g,c} = \mathsf{T}$ or $[\![\psi]\!]^{M,g,c} = \mathsf{T}$.

   (c) $[\![\phi \rightarrow \psi]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g,c} = \mathsf{F}$ or $[\![\psi]\!]^{M,g,c} = \mathsf{T}$.

   (d) $[\![\phi \leftrightarrow \psi]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\phi]\!]^{M,g,c} = [\![\psi]\!]^{M,g,c}$.

6. **Quantification**

   (a) $[\![\forall u\,.\,\phi]\!]^{M,g,c} = \mathsf{T}$ iff for all $o \in D_\tau$: $[\![\phi]\!]^{M,g[u\mapsto o],c} = \mathsf{T}$ (where τ is the type of u).

   (b) $[\![\exists u\,.\,\phi]\!]^{M,g,c} = \mathsf{T}$ iff there exists $o \in D_\tau$ such that $[\![\phi]\!]^{M,g[u\mapsto o],c} = \mathsf{T}$.

7. **Lambda Abstraction**
   $[\![\lambda u\,.\,\alpha]\!]^{M,g,c}$ is that function f from D_σ to D_τ such that for all $o \in D_\sigma$: $f(o) = [\![\alpha]\!]^{M,g[u\mapsto o],c}$ (where u has type σ and α has type τ).

8. **Temporal Ordering**
   $[\![\alpha < \beta]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\alpha]\!]^{M,g,c} < [\![\beta]\!]^{M,g,c}$;
   $[\![\alpha \subseteq \beta]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\alpha]\!]^{M,g,c} \subseteq [\![\beta]\!]^{M,g,c}$.

9. **Mereological Part-Of**
   $[\![\alpha \sqsubseteq \beta]\!]^{M,g,c} = \mathsf{T}$ iff $[\![\alpha]\!]^{M,g,c} \sqsubseteq [\![\beta]\!]^{M,g,c}$.
   The terms $\alpha \oplus \beta$ and $\bigoplus \pi$ denote the binary sum and generalized sum respectively, as uniquely determined by $\sqsubseteq$.

10. **Indexical Constants**
    The four indexical constants take their values from the context:

    - $[\![\mathrm{i}]\!]^{M,g,c} = \mathsf{sp}(c)$ (type e: speaker)

- $[\![\text{u}]\!]^{M,g,c} = \text{ad}(c)$ (type e: addressee)
- $[\![\text{now}]\!]^{M,g,c} = \text{time}(c)$ (type i: speech time)
- $[\![\text{here}]\!]^{M,g,c} = \text{loc}(c)$ (type e: location)

11. **Presupposition**
$[\![\partial(\phi)]\!]^{M,g,c} = \top$ if $[\![\phi]\!]^{M,g,c} = \top$; undefined otherwise.

13.4 Unified fragment

In this section, we present a unified fragment that weaves together all of the developments along independent frontiers from the preceding chapters into a single coherent system. There is some groundwork that must be laid first in order for this to be accomplished.

One choice to make is whether to use event-predicate approach or the event-quantifier approach. We have opted for the event-quantifier approach here (Chapter 10, Extension B).

Second, in order to link up event semantics with tense and aspect, we must associate each event with its runtime. We do this using the runtime function τ; $\tau(e)$ denotes the 'temporal trace' of event e, the time interval during which e takes place.

In order to incorporate both tense and modality, we use the tense–aspect–modality architecture of Sections 13.1 and 13.3. Including events in the system makes it necessary to connect events to worlds as well as time. We use $\chi(e)$ to pick out the world in which e takes place. This function figures in the rule of Event Quantification (EQ) in the fragment below.

The analysis of propositional attitude verbs in an event-semantic framework, developed in Section 13.2, is incorporated in the fragment below.

13.4.1 The fragment

Representation language.
The full Ty2$^+$ system with six primitive types: e (individuals), i

(times), v (events), s (worlds), n (natural numbers), t (truth values). Formal definitions in Section 13.3.

Variable conventions.
The formal variable for type τ and index n is $v_{n,\tau}$. Subscripted forms x_i, t_j, e_j abbreviate $v_{i,e}$, $v_{j,i}$, $v_{j,v}$ respectively (used when the index carries semantic content, as in pronoun and trace translation). All other variable symbols listed below stand for arbitrary variables of the indicated type; when multiple such symbols appear in the same formula, they are understood to be distinct from one another and from any subscripted form in the same formula (distinct as variables, not necessarily as values).

- Type e: x, y, z, x_i
- Type i: t, t', t'', t_n
- Type $\langle i,t\rangle$: T, U
- Type v: e, e', e''
- Type s: w_0, w, w'
- Type $\langle e,t\rangle$: P, Q
- Type $\langle e,\langle e,t\rangle\rangle$: R
- Type $\langle\langle e,t\rangle,t\rangle$: Z
- Type $\langle v,t\rangle$: V, V'
- Type $\langle\langle v,t\rangle,t\rangle$: $\mathcal{Q}$
- Type $\langle s,t\rangle$: p, q

Note: in Chapters 6–11, p and q ranged over type t (truth values). From Chapter 12 onward, they range over type $\langle s,t\rangle$ (propositions), reflecting the shift to an intensional language.
Additional variable symbols can be constructed by adding an explicit type subscript (e.g., $P_{\langle i,st\rangle}$).

Indexical constants.
Value determined by the context of utterance c:

- i (type e): speaker, $\mathsf{sp}(c)$
- u (type e): addressee, $\mathsf{ad}(c)$
- now (type i): speech time, $\mathsf{time}(c)$
- here (type e): location, $\mathsf{loc}(c)$

Special constants.

- τ (type $\langle v, i\rangle$), χ (type $\langle v, s\rangle$)
- agent, theme, source, goal, exp, stim (each type $\langle v, e\rangle$)
- cont (type $\langle e, st\rangle$, mapping content objects to their propositional content)

Accessibility relations.

- deon, epist of type $\langle\langle s, \langle i, \langle s, t\rangle\rangle\rangle, t\rangle$: flavor predicates of accessibility relations; $\mathcal{R}^{\mathsf{f}}$ abbreviates $\mathcal{R}_{|\mathsf{f}(\mathcal{R})}$ (see Section 12.5.1)
- dox, boul of type $\langle\langle e, \langle s, \langle i, \langle s, t\rangle\rangle\rangle\rangle, t\rangle$: flavor predicates of agent-parameterized accessibility relations, appearing in the meaning postulates for believe and want events (see below)

Syntax.

$$
\begin{array}{lcl}
\text{TP} & \rightarrow & \text{T}' \ (\text{Asp}_{\text{Ord}}\text{P} \mid \text{ModP}) \\
\text{T}' & \rightarrow & \text{TENSE T}_n \\
\text{Asp}_{\text{Ord}}\text{P} & \rightarrow & \text{Asp}_{\text{Ord}} \ \text{Asp}_{\text{Inc}}\text{P} \\
\text{Asp}_{\text{Inc}}\text{P} & \rightarrow & \text{Asp}_{\text{Inc}} \ \text{vP}
\end{array}
$$

ModP	→	Mod Asp_{Ord}P
VP	→	V (DP|AP|PP|CP)
CP	→	(C) TP
CP	→	DP_i[WH] TP
AP	→	A (PP)
DP	→	D (NP)
DP	→	DP[GEN] NP
DP[GEN]	→	DP Poss
NP	→	N (PP)
N	→	N PL
NP	→	A NP
NP	→	NP CP
PP	→	P DP
DP	→	θ DP
TP	→	AdvP TP
TP	→	SNeg TP
VP	→	VP AdvP
AdvP	→	Adv

Schemas (where *XP* ranges over any phrasal category):

XP	→	Neg *XP*
XP	→	*XP* &*XP*
&*XP*	→	& *XP*

Syntactic transformations.

Two syntactic transformations apply:

- **Relative clause formation**: a DP_i[WH] moves to the specifier of CP, leaving a trace t_i.
- **Quantifier Raising (QR)**: a quantificational DP may adjoin to a type t node at LF, leaving a trace t_i; interpreted via Predicate Abstraction.

Composition rules.

- **Function Application (FA)**: If $\alpha \rightsquigarrow \alpha'$ of type $\langle \sigma, \tau \rangle$ and $\beta \rightsquigarrow \beta'$ of type σ, then $[\alpha\ \beta] \rightsquigarrow \alpha'(\beta')$.
- **Predicate Modification (PM)**: If $\alpha \rightsquigarrow \alpha'$ and $\beta \rightsquigarrow \beta'$, both of type $\langle e, t \rangle$ (or both of type $\langle v, t \rangle$), then $[\alpha\ \beta] \rightsquigarrow \lambda u.\, \alpha'(u) \wedge \beta'(u)$.
- **Predicate Abstraction (PA)**: If $\beta \rightsquigarrow \beta'$, then $[\lambda_i\ \beta] \rightsquigarrow \lambda x_i.\, \beta'$.
- **Pronouns and Traces Rule**: If α_i is an indexed pronoun or trace, then $\alpha_i \rightsquigarrow x_i$.
- **Intensional Function Application (IFA)**: If $\alpha \rightsquigarrow \alpha'$ of type $\langle \langle s, \sigma \rangle, \tau \rangle$ and $\beta \rightsquigarrow \beta'$ of type σ, then $[\alpha\ \beta] \rightsquigarrow \alpha'(\lambda w.\, \beta')$ (see Section 12.4.4.2).

Type-shifting operations.

From Chapter 9:

- **Individual Raising**: if α translates as α' of type e, then α also has a reading of type $\langle \langle e, t \rangle, t \rangle$: $\lambda P.\, P(\alpha')$

From Chapter 7:

- **Object Raising** (RAISE-O): if α translates as α' of type $\langle e, \langle a, t \rangle \rangle$, then α also has a reading of type $\langle \langle \langle e, t \rangle, t \rangle, \langle a, t \rangle \rangle$:

$$\lambda Z \lambda x_a.\, Z(\lambda y.\, \alpha'(y)(x))$$

- **Subject Raising** (RAISE-S): if α translates as α' of type $\langle a, \langle e, t \rangle \rangle$, then α also has a reading of type $\langle a, \langle \langle \langle e, t \rangle, t \rangle, t \rangle \rangle$:

$$\lambda y_a \lambda Z.\, Z(\lambda x.\, \alpha'(y)(x))$$

From Chapter 10 (Extension B):

- **Individual θ-Participant** (IND+): if α translates as θ of type $\langle v, e\rangle$, then α also has a reading of type $\langle e, \langle\langle\langle v, t\rangle, t\rangle, \langle\langle v, t\rangle, t\rangle\rangle\rangle$:

$$\lambda x \lambda \mathcal{Q} \lambda V . \mathcal{Q}(\lambda e . \theta(e) = x \wedge V(e))$$

- **Predicate θ-Participant** (PRED+): if α translates as θ of type $\langle v, e\rangle$, then α also has a reading of type $\langle\langle e, t\rangle, \langle\langle\langle v, t\rangle, t\rangle, \langle\langle v, t\rangle, t\rangle\rangle\rangle$:

$$\lambda P \lambda \mathcal{Q} \lambda V . \mathcal{Q}(\lambda e . P(\theta(e)) \wedge V(e))$$

New in this chapter:

- **Event Quantification** (EQ): if α translates as $\mathcal{Q}'$ of type $\langle\langle v, t\rangle, t\rangle$, then α also has a reading of type $\langle i, st\rangle$:

$$\lambda t . \lambda w . \mathcal{Q}'(\lambda e . \tau(e) = t \wedge \chi(e) = w)$$

  EQ applies at vP, converting an event-quantifying denotation into a property of time–world pairs (type $\langle i, st\rangle$). It replaces Quantifier Closure (QC) from Chapter 10: rather than closing off the event quantifier into a truth value, EQ feeds the tense–aspect machinery above vP.

Lexical entries.

N:

- Intransitive nouns follow $W \rightsquigarrow \lambda x . W(x)$.
- Relational nouns follow $W \rightsquigarrow \lambda y . \lambda x . W(x, y)$ (type $\langle e, \langle e, t\rangle\rangle$).

A (type $\langle e, t\rangle$):

- Adjectives follow $W \rightsquigarrow \lambda x . W(x)$.

D (proper names, type e):

- Proper names translate to type-e constants (e.g., *Alex* $\rightsquigarrow$ a).

D (determiners) and numerals:

- *a/an* $\rightsquigarrow \lambda P.P$
  (type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$; used in predicative position)
- *some* $\rightsquigarrow \lambda P\lambda Q.\exists x[P(x)\wedge Q(x)]$
  (type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$)
- *every* $\rightsquigarrow \lambda P\lambda Q.[\partial(\exists x.P(x))\wedge\forall x[P(x)\rightarrow Q(x)]]$
  (type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$)
- *no* $\rightsquigarrow \lambda P\lambda Q.\neg\exists x[P(x)\wedge Q(x)]$
  (type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$)
- $the_{\text{SUPR}} \rightsquigarrow \lambda P.\iota x[P(x)\wedge\forall y[P(y)\rightarrow y\sqsubseteq x]]$
  (type $\langle\langle e,t\rangle,e\rangle$)
- $\varnothing_D \rightsquigarrow \lambda P\lambda Q.\exists x[P(x)\wedge Q(x)]$
  (type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$; null indefinite D)
- Cardinals: $n \rightsquigarrow \lambda x.\mathsf{card}(x)=n$
  (type $\langle e,t\rangle$; e.g., *two, three,* ...)

D (quantificational noun phrases):

- *somebody* $\rightsquigarrow \lambda P.\exists x[\mathsf{person}(x)\wedge P(x)]$
- *nobody* $\rightsquigarrow \lambda P.\neg\exists x[\mathsf{person}(x)\wedge P(x)]$
- *everybody* $\rightsquigarrow \lambda P.[\partial(\exists x.\mathsf{person}(x))\wedge\forall x[\mathsf{person}(x)\rightarrow P(x)]]$
- *something* $\rightsquigarrow \lambda P.\exists x.P(x)$
- *nothing* $\rightsquigarrow \lambda P.\neg\exists x.P(x)$
- *everything* $\rightsquigarrow \lambda P.\forall x.P(x)$

Poss:

- *'s* $\rightsquigarrow \lambda y\lambda R.\lambda x.R(y)(x)$
  (type $\langle e,\langle\langle e,\langle e,t\rangle\rangle,\langle e,t\rangle\rangle\rangle$)

P:

- *of* $\rightsquigarrow \lambda x.x$
  (type $\langle e,e\rangle$)
- *with* $\rightsquigarrow \lambda y\lambda x.\mathsf{with}(x,y)$
  (type $\langle e,\langle e,t\rangle\rangle$)

Neg:

- *not* $\rightsquigarrow \lambda\mathcal{Q}\lambda V.\neg\mathcal{Q}(V)$
  (type $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle$)

SNeg:

- *It is not the case that* $\rightsquigarrow \lambda p.\neg p$
  (type $\langle st,st\rangle$)

Plurals and coordination:

- PL $\rightsquigarrow \lambda P.{}^{*}P$
  (type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$)
- $and_{DP} \rightsquigarrow \lambda Q'\lambda Q\lambda P.Q(P)\wedge Q'(P)$
  (type $\langle\langle\langle e,t\rangle,t\rangle,\langle\langle\langle e,t\rangle,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle\rangle$)
- $or_{DP} \rightsquigarrow \lambda Q'\lambda Q\lambda P.Q(P)\vee Q'(P)$
  (type $\langle\langle\langle e,t\rangle,t\rangle,\langle\langle\langle e,t\rangle,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle\rangle$)
- $and_{VP} \rightsquigarrow \lambda\mathcal{Q}'\lambda\mathcal{Q}\lambda V.\mathcal{Q}(V)\wedge\mathcal{Q}'(V)$
  (type $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$)
- $or_{VP} \rightsquigarrow \lambda\mathcal{Q}'\lambda\mathcal{Q}\lambda V.\mathcal{Q}(V)\vee\mathcal{Q}'(V)$
  (type $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$)

The and_S/or_S entries (type $\langle t,\langle t,t\rangle\rangle$) and the and_{VP}/or_{VP} entries above follow the Partee–Rooth schema. The and_{VP}/or_{VP} entries are specific to the event-quantifier approach adopted here; $\mathcal{Q}$ and V are the event-quantifier and event-predicate variables respectively.

θ-role heads (type $\langle v, e\rangle$):

- AGENT $\rightsquigarrow$ agent
- THEME $\rightsquigarrow$ theme
- SOURCE $\rightsquigarrow$ source
- GOAL $\rightsquigarrow$ goal
- EXP $\rightsquigarrow$ exp
- STIM $\rightsquigarrow$ stim

V (eventive and stative, type $\langle\langle v, t\rangle, t\rangle$):
Schema: $W \rightsquigarrow \lambda V.\exists e.W(e) \wedge V(e)$.
Thematic arguments are introduced by θ-role heads via IND+ or PRED+; the verb itself carries no argument structure.

V (propositional attitude verbs, type $\langle\langle v, t\rangle, t\rangle$):
Following the content-object analysis of Section 13.2, attitude verbs are predicates of attitude events; the propositional content is contributed by the THEME θ-head (via PRED+) applied to a Cont-CP:

- *believe*/*believes* $\rightsquigarrow \lambda V.\exists e.\mathsf{believe}(e) \wedge V(e)$
- *want*/*wants* $\rightsquigarrow \lambda V.\exists e.\mathsf{want}(e) \wedge V(e)$

The EXP θ-head introduces the attitude-holder via IND+; the THEME θ-head introduces the content object. Doxastic/bouletic accessibility is a model-theoretic constraint on what events count as believing/wanting events.

Asp_{Inc} (type $\langle\langle i, st\rangle, \langle i, st\rangle\rangle$):

- PFV $\rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\exists t''[t'' \subseteq t \wedge P(t'')(w)]$
- PROG $\rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\exists t''[t \subseteq t'' \wedge P(t'')(w)]$

Asp$_{\text{Ord}}$ (type $\langle\langle i,st\rangle,\langle i,st\rangle\rangle$):

- NPERF $\rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.P(t)(w)$
- PERF $\rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\exists t'[t' < t \wedge P(t')(w)]$
- WOLL $\rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\exists t'[t < t' \wedge P(t')(w)]$

Mod (type $\langle\langle i,st\rangle,\langle i,st\rangle\rangle$):

- *must*$_{\text{EPIST}} \rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\forall w'[\mathcal{R}^{\text{epist}}_{w,t}(w') \rightarrow P(t)(w')]$
- *have to*$_{\text{EPIST}} \rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\forall w'[\mathcal{R}^{\text{epist}}_{w,t}(w') \rightarrow P(t)(w')]$
- *might*$_{\text{EPIST}} \rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\exists w'[\mathcal{R}^{\text{epist}}_{w,t}(w') \wedge P(t)(w')]$
- *possibly*$_{\text{EPIST}} \rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\exists w'[\mathcal{R}^{\text{epist}}_{w,t}(w') \wedge P(t)(w')]$
- *have to*$_{\text{DEON}} \rightsquigarrow \lambda P_{\langle i,st\rangle}.\lambda t.\lambda w.\forall w'[\mathcal{R}^{\text{deon}}_{w,t}(w') \rightarrow P(t)(w')]$
  (free $\mathcal{R}^{\text{deon}}$, type $\langle s,\langle i,\langle s,t\rangle\rangle\rangle$)

T$_n$ (type i): the reference time, supplied by the context.

TENSE (type $\langle i,i\rangle$):

- PAST $\rightsquigarrow \lambda t.t^{\text{past}}$
- PRES $\rightsquigarrow \lambda t.t^{\text{pres}}$

C: Two readings of *that*:

- *that* (transparent, type $\langle st,st\rangle$) $\rightsquigarrow \lambda p_{st}.p$
  (used in modal and tense contexts)
- *that* (Cont-CP, type $\langle st,\langle e,t\rangle\rangle$) $\rightsquigarrow \lambda p_{st}.\lambda x.\text{cont}(x) = p$
  (used in attitude verb contexts)

Syncategorematically interpreted items.

The following have no independent denotation; their interpretation is determined entirely by the composition rules above:

D[WH] (relative pronouns): *who, which* — interpreted via Predicate Abstraction

Dem[SIM]: *such* — interpreted via Predicate Abstraction

D (personal pronouns and traces): *he, she, him, her, it, they, them*, traces t_i — interpreted as x_i via the Pronouns and Traces Rule

Meaning postulates.

The following meaning postulates constrain what events count as believing and wanting events by linking the event structure to a doxastic or bouletic accessibility relation:

- $\forall e[\mathsf{believe}(e) \rightarrow \exists \mathcal{R}[\mathsf{dox}(\mathcal{R}) \wedge \forall w'[\mathcal{R}(\mathsf{exp}(e))(\chi(e))(\tau(e))(w') \rightarrow \mathsf{cont}(\mathsf{theme}(e))(w')]]]$
- $\forall e[\mathsf{want}(e) \rightarrow \exists \mathcal{R}[\mathsf{boul}(\mathcal{R}) \wedge \forall w'[\mathcal{R}(\mathsf{exp}(e))(\chi(e))(\tau(e))(w') \rightarrow \mathsf{cont}(\mathsf{theme}(e))(w')]]]$

Bibliography

Abusch, Dorit. 1988. Sequence of tense, intensionality and scope. In Hagit Borer (ed.), *Proceedings of WCCFL 7*, 1–14.

Abusch, Dorit. 1997. Sequence of tense and temporal *de re*. *Linguistics and Philosophy* 20(1). 1–50.

Ahn, Dorothy & Uli Sauerland. 2015. The grammar of relative measurement. In *Semantics and Linguistic Theory (SALT) 25*, 125–142. doi:10.3765/salt.v25i0.3062.

Anderson, Jill. 2014. Misreading like a lawyer: Cognitive bias in statutory interpretation. *Harvard Law Review* 1521. 1563–68.

Anderson, Jill. 2019. Logic problems. Manuscript, University of Connecticut.

Bach, Emmon. 1986. The algebra of events. *Linguistics and Philosophy* 15. 5–16.

Barker, Chris. 2005. Remark on Jacobson 1999: Crossover as a local constraint. *Linguistics and Philosophy* 28(4). 447–472.

Barker, Chris & Chung-chieh Shan. 2014. *Continuations and natural langauge*. Oxford, UK: Oxford University Press.

Barwise, Jon & Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy* 4. 159–219.

Bayer, Joseph. 1984. Towards an explanation of certain that-i phenomena: The COMP-node in Bavarian. In Wim de Geest & Yvan Putseys (eds.), *Sentential complementation*, 23–32. Dordrecht, Netherlands: De Gruyter. doi:10.1515/9783110882698-004.

Beaver, David & Emiel Krahmer. 2001. A partial account of presupposition projection. *Journal of Logic, Language and Information* 10. 147–182.

Bennett, Jonathan. 2003. *A philosophical guide to conditionals*. Oxford, UK: Oxford University Press. doi:10.1093/0199258872.001.0001.

Bennett, Michael & Barbara Partee. 1972. Toward the logic of tense and aspect in English. Ms., System Development Corporation.

Bernard, Timothée & Lucas Champollion. 2018. Negative events in compositional semantics. *Semantics and Linguistic Theory* 28. 512. doi:10.3765/salt.v28i0.4429. `https://doi.org/10.3765/salt.v28i0.4429`.

Bhatt, Rhajesh & Roumyana Pancheva. 2005. Lsa 130: The syntax and semantics of aspect. Lecture notes from the 2005 LSA Institute.

Boas, Hans C. & Ivan A. Sag (eds.). 2012. *Sign-based construction grammar*. University of Chicago Press.

Bochnak, Ryan. 2016. Past time reference in a language with optional tense. *Linguistics and Philosophy* 39. 247–294.

Bondarenko, Tatiana. 2022. *Anatomy of an attitude*: MIT dissertation.

Bresnan, Joan. 2001. *Lexical-functional syntax*. Malden, MA: Blackwell.

Bumford, Dylan & Simon Charlow. 2026. *Effect-driven interpretation: Functors for natural language composition* Elements in Semantics. Cambridge University Press.

Cable, Seth. 2008. Tense, aspect and aktionsart. Lecture notes, Theoretical Perspectives on Languages of the Pacific Northwest, Proseminar on Semantic Theory.

Cable, Seth. 2017. The implicatures of optional past tense in Tlingit and the implications for 'discontinuous past'. *Natural Language and Linguistic Theory* 35(3). 635–681.

Carlson, Gregory N. 1977. *Reference to kinds in English*. Amherst, MA: University of Massachusetts dissertation.

Carlson, Gregory N. 1984. Thematic roles and their role in semantic interpretation. *Linguistics* 22(3). 259–280. doi:10.1515/ling.1984.22.3.259.

Carnie, Andrew. 2013. *Syntax: A generative introduction*. Blackwell.

Carpenter, Bob. 1998. *Type-logical semantics*. MIT Press.

Castañeda, Hector-Neri. 1967. Comments. In Nicholas Rescher (ed.), *The logic of decision and action*, University of Pittsburgh Press.

Chalmers, David J. 2006. The foundations of two-dimensional semantics. In Josep Macià & Manuel García-Carpintero (eds.), *Two-dimensional semantics*, chap. 4, 55–140. Oxford University Press. doi:10.1093/oso/9780199271955.003.0004.

Champollion, Lucas. 2014. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy* 38(1). 31–66.

Champollion, Lucas. 2017. *Parts of a whole: Distributivity as a bridge between aspect and measurement*. Oxford University Press. doi:10.1093/oso/9780198755128.001.0001.

Champollion, Lucas & Manfred Krifka. 2016. Mereology. In Paul Dekker & Maria Aloni (eds.), *Cambridge handbook of semantics*, Cambridge University Press. `http://ling.auf.net/lingbuzz/002099`.

Champollion, Lucas, Josh Tauberer & Maribel Romero. 2007. The penn lambda calculator: Pedagogical software for natural language semantics. In Tracy Holloway King & Emily M. Bender (eds.), *Proceedings of the grammar engineering across frameworks (geaf) 2007 workshop*, CSLI On-line Publications.

Chierchia, Gennaro. 1998. Reference to kinds across languages. *Natural Language Semantics* 6. 339–405.

Chierchia, Gennaro & Sally McConnell-Ginet. 2000. *Meaning and grammar: An introduction to semantics*. Cambridge, MA: MIT Press 2nd edn.

Chomsky, Noam. 1957. *Syntactic structures*. The Hague: Mouton.

Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Chomsky, Noam. 1973. Conditions on transformations. In Stephen Anderson & Paul Kiparsky (eds.), *A festschrift for Morris Halle*, 232–286. New York: Holt, Rinehart and Winston.

Chomsky, Noam. 1995. *The minimalist program*. Cambridge, MA: MIT Press.

Chomsky, Noam & Howard Lasnik. 1977. Filters and control. *Linguistic Inquiry* 8. 435–504.

Comrie, Bernard. 1976. *Aspect*. Cambridge: Cambridge University Press.

Condoravdi, Cleo. 2002. Temporal interpretation of modals: Modals for the present and for the past. In David Beaver, Stefan Kaufmann, Brady Clark & Luis Casillas (eds.), *Stanford papers on semantics*, 59–88. Stanford: CSLI Publications.

Cooper, Robin. 1983. *Quantification and syntactic theory*. Reidel.

Cresswell, Max J. 2012. *Entities and indices* Studies in Linguistics and Philosophy. Dordrecht, Netherlands: Kluwer.

Croft, William A. & David Alan Cruse. 2004. *Cognitive linguistics*. Cambridge, UK: Cambridge University Press. doi:10.1017/CBO9780511803864.

Curry, Haskell B. & Robert Feys. 1958. *Combinatory logic*, vol. 1. Amsterdam: North-Holland.

Davidson, Donald. 1967. The logical form of action sentences. In Nicholas Rescher (ed.), *The logic of decision and action*, 81–95. Pittsburgh, PA: University of Pittsburgh Press. doi:10.1093/0199246270.003.0006.

Dowty, David. 1977. Toward a semantic analysis of verb aspect and the English 'imperfective' progressive. *Linguistics and Philosophy* 1(1). 45–77.

Dowty, David, Robert E. Wall & Stanley Peters. 1981. *Introduction to Montague semantics*. Dordrecht: Kluwer.

Dowty, David R. 1979. *Word meaning and Montague grammar: The semantics of verbs and times in generative semantics and in Montague's PTQ*, vol. 7 Studies in Linguistics and Philosophy. Dordrecht, Netherlands: Reidel. doi:10.1007/978-94-009-9473-7.

Doxiadis, Apostolos & Christos H. Papadimitriou. 2009. *Logicomix: An epic search for truth*. New York: Bloomsbury.

Evans, Gareth. 1977. Pronouns, quantifiers, and relative clauses. *Canadian Journal of Philosophy* 7.

Fauconnier, Gilles. 1975. Pragmatic scales and logical structure. *Linguistic Inquiry* 6(3). 353–375.

von Fintel, Kai. 1999. NPI licensing, Strawson entailment, and context-dependency. *Journal of Semantics* 16. 97–148.

von Fintel, Kai. 2006. Modality and language. In Donald M. Borchert (ed.), *Encyclopedia of philosophy – second edition*, Detroit: MacMillan Reference USA.

von Fintel, Kai. 2011. Conditionals. In Klaus von Heusinger, Claudia Maienborn & Paul Portner (eds.), *Semantics: An international handbook of natural language meaning*, vol. 3 Handbücher zur Sprach- und Kommunikationswissenschaft / Handbooks of Linguistics and Communication Science (HSK), chap. 59, 1515–1538. de Gruyter. doi:10.1515/9783110255072.

Fox, Danny. 2002. Antecedent-contained deletion and the copy theory of movement. *Linguistic Inquiry* 33(1). 63–96. doi:10.1162/002438902317382189.

Frege, Gottlob. 1892. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik* 100. 25–50. Translated as *On Sense and Reference* by M. Black in *Translations from the Philosophical Writings of Gottlob Frege*, P. Geach and M. Black (eds. and trans.), Oxford: Blackwell, third edition, 1980.

Frege, Gottlob. 1983. Grundgesetze der Arithmetik. Verlag Hermann Pohle, Jena. Reprinted 1962 by Georg Olms, Hildesheim, Germany and 1966 as No. 32 in *Olms Paperbacks* series.

Gallin, Daniel. 1975. *Intensional and higher order modal logic*. Amsterdam: North Holland Press.

Geurts, Bart. 2011. *Quantity implicatures*. Cambridge: Cambridge University Press.

Goranko, Valentin & Antje Rumberg. 2023. Temporal logic. In Edward N. Zalta & Uri Nodelman (eds.), *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University.

Grice, Paul. 1975. Logic and conversation. In Peter Cole & Jerry Morgan (eds.), *Syntax and semantics*, vol. 3, 41–58. New York: Academic Press.

Groenendijk, Jeroen, Theo Janssen & Martin Stokhof (eds.). 1984. *Truth, interpretation and information: selected papers from the third amsterdam colloquium*. Dordrecht, Netherlands: Foris.

Gruber, Jeffrey S. 1965. *Studies in lexical relations*. Cambridge, MA: Massachusetts Institute of Technology dissertation. `http://hdl.handle.net/1721.1/13010`.

Hackl, Martin. 2009. On the grammar and processing of proportional quantifiers: *most* vs. *more than half*. *Natural Language Semantics* 17. 63–98.

Halle, Morris & Alec Marantz. 1993. Distributed morphology and the pieces of inflection. *The view from building* 20. 111–176.

Haug, Dag. 2013. Partial dynamic semantics for anaphora: Compositionality without syntactic coindexation. *Journal of Semantics* (online first).

Heim, Irene. 1982. *The semantics of definite and indefinite noun phrases*: U. Mass Amherst dissertation.

Heim, Irene. 1983. On the projection problem for presuppositions. In Daniel Flickinger, Michael Barlow & Michael Westcoat (eds.), *Proceedings of the second west coast conference on formal linguistics*, 114–125. Stanford, CA: Stanford University Press.

Heim, Irene. 1994. Comments on Abusch's theory of tense. In Hans Kamp (ed.), *Ellipsis, tense and questions*, DYANA Deliverable R2.2B.

Heim, Irene & Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Hendriks, Hermann. 1993. *Studied flexibility*: ILLC dissertation.

Higginbotham, James. 1983. The logic of perceptual reports: An extensional alternative to situation semantics. *The Journal of Philosophy* 80(2). 100–127. doi:10.2307/2026237.

Hindley, J. Roger & Jonathan P. Seldin. 2008. *Lambda-calculus and combinators: An introduction*. Cambridge: Cambridge University Press.

Hintikka, Jaako. 1969. Semantics for propositional attitudes. In J. W. Davis, D. J. Hockney & W. K. Wilson (eds.), *Philosophical logic*, 21–45. Dordrecht: Reidel.

Horn, Laurence R. 1972. On the semantic properties of logical operators in English. Doctoral dissertation, University of California, Los Angeles.

Horn, Laurence R. 1985. Metalinguistic negation and pragmatic ambiguity. *Language* 61(1). 121–174. doi:10.2307/413423.

Horn, Lawrence. 2018. Contradiction. In *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University Winter 2018 edn.

Huang, Shuan-Fan. 1981. On the scope phenomena of Chinese quantifiers. *Journal of Chinese Linguistics* 9(2). 226–243.

Hughes, G.E. & M. J. Cresswell. 1968. *An introduction to modal logic*. London: Methuen and Co Ltd.

Iatridou, Sabine, Elena Anagnostopoulou & Roumyana Izvorski. 2001. Observations about the form and meaning of the Perfect. In *Ken Hale: A life in language*, 189–238. Cambridge, MA: MIT Press. doi:10.1515/9783110902358.153.

Jackendoff, Ray S. 1972. *Semantic interpretation in generative grammar*. Cambridge, MA: MIT Press.

Jacobson, Pauline. 1992. Antecedent Contained Deletion in a Variable-Free Semantics. *Semantics and Linguistic Theory* 2. doi:10.3765/salt.v2i0.3027. `https://journals.linguisticsociety.org/proceedings/index.php/SALT/article/view/3027`.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22. 117–184.

Jacobson, Pauline. 2000. Paycheck pronouns, Bach-Peters sentences, and variable-free semantics. *Natural Language Semantics* 8. 77–155.

Jacobson, Pauline. 2012. Direct compositionality. In *The Oxford handbook of compositionality*, 109–129. Oxford University Press.

Jespersen, Otto. 1914/1970. *A Modern English grammar on historical principles. part II: Syntax*. London and Copenhagen: George Allen & Unwin and Ejnar Munksgaard.

Kamp, Hans. 1971. Formal properties of 'now'. *Theoria* 37(3). 227–273.

Kamp, Hans & Uwe Reyle. 1993. *From discourse to logic*. Dordrecht: Kluwer Academic Publishers.

Kaplan, David. 1977. Demonstratives: An essay on the semantics, logic, metaphysics, and epistemology of demonstratives and other indexicals. In Joseph Almog, John Perry & Howard

Wettstein (eds.), *Themes from Kaplan*, 267–298. Oxford: Oxford University Press.

Keenan, Edward L. & Jonathan Stavi. 1986. A semantic characterization of natural language determiners. *Linguistics and Philosophy* 9. 253–326.

Kennedy, Christopher. 2015. A 'de-Fregean' semantics (and neo-Gricean pragmatics) for modified and unmodified numerals. *Semantics and Pragmatics* 8(10). 1–44. doi:10.3765/sp.8.10.

Kipper-Schuler, Karin. 2005. *Verbnet: A broad-coverage, comprehensive verb lexicon*. Philadelphia, PA: University of Pennsylvania dissertation.

Klein, Wolfgang. 1994. *Time in language*. London and New York: Routledge.

Kratzer, Angelika. 1998. More structural analogies between pronouns and tense. In Devon Strolovitch & Aaron Lawson (eds.), *SALT VIII: Proceedings of the second conference on semantics and linguistic theory*, 92–110. Ithaca, NY: CLC Publications.

Kratzer, Angelika. 2016. Evidential moods in attitude and speech reports. Slides presented at the University of Connecticut, the University of Pennsylvania, and the 1st Syncart Workshop, University of Siena.

Krifka, Manfred. 1986. *Nominalreferenz und Zeitkonstitution. Zur Semantik von Massentermen, Pluraltermen und Aspektklassen*. Munich, Germany (published 1989): Fink.

Krifka, Manfred. 1989. Nominal reference, temporal constitution and quantification in event semantics. In Renate Bartsch, Johan van Benthem & Peter van Emde Boas (eds.), *Semantics and contextual expression*, 75–115. Dordrecht, Netherlands: Foris.

Krifka, Manfred. 1992. Thematic relations as links between nominal reference and temporal constitution. In Ivan A. Sag & Anna Szabolcsi (eds.), *Lexical matters*, 29–53. Stanford, CA: CSLI Publications.

Krifka, Manfred. 1998. The origins of telicity. In Susan D. Rothstein (ed.), *Events and grammar*, Dordrecht: Kluwer Academic Publishers.

Kripke, Saul. 1963. Semantical considerations on modal logic. *Acta Philosophica Fennica* 16. 83–89.

Kripke, Saul. 1979. A puzzle about belief. In *Meaning and use*, 239–283. Dordrecht: Reidel.

Kripke, Saul. 1980. *Naming and necessity*. Harvard University Press.

Kroch, Anthony S. 1974. *The semantics of scope in English*. Cambridge, MA: Massachusetts Institute of Technology dissertation.

Ladusaw, William A. 1980. On the notion 'affective' in the analysis of negative polarity items. *Journal of Linguistic Research* 1. 1–16.

Landman, Fred. 2000. *Events and plurality: The Jerusalem lectures*, vol. 76 Studies in Linguistics and Philosophy. Dordrecht, Netherlands: Kluwer. doi:10.1007/978-94-011-4359-2.

Landman, Fred. 2004. *Indefinites and the type of sets*. Malden, MA: Blackwell.

LaPierre, Serge. 1992. A functional partial semantics for Intensional Logic. *Notre Dame Journal of Formal Logic* 33(4). 517–541.

Lasnik, Howard & Terje Lohndal. 2013. Brief overview of the history of generative syntax. In *The Cambridge handbook of generative syntax*, 26–60. Cambridge: Cambridge University Press.

Levin, Beth. 1993. *English verb classes and alternations: A preliminary investigation.* Chicago, IL: University of Chicago Press.

Lewis, David. 1979. Scorekeeping in a language game. *Journal of Philosophical Logic* 8. 339–359.

Link, Godehard. 1983. The logical analysis of plurals and mass terms: A lattice-theoretical approach. In Rainer Bäuerle, Christoph Schwartze & Arnim von Stechow (eds.), *Meaning, use, and the interpretation of language*, 302–323. Berlin: Walter de Gruyter.

Matthewson, Lisa. 2006. Temporal semantics in a superficially tenseless language. *Linguistics and Philosophy* 29. 673–713.

May, Robert. 1985. *Logical form: Its structure and derivation.* MIT Press.

McCoard, Robert W. 1979. *The English perfect: Tense-choice and pragmatic inferences.* Amsterdam: North-Holland Press.

Menzel, Christopher. 2018. Actualism. In Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University summer 2018 edn. `https://plato.stanford.edu/archives/sum2018/entries/actualism/`.

Milsark, Gary. 1977. Toward an explanation of certain peculiarities of the existential construction in English. *Linguistic Analysis* 3. 1–29.

Montague, Richard. 1970. English as a formal language. In Bruno Visentini & Camillo Olivetti (eds.), *Linguaggi nella societá e nella tecnica*, vol. 87 Saggi di cultura contemporanea, 188–221. Edizioni di Comunitá.

Montague, Richard. 1973a. Comments on Moravcsik's paper. In K.J.J. Hintikka, J.M.E. Moravcsik & P. Suppes (eds.), *Approaches to natural language: Proceedings of the 1970 Stanford workshop on grammar and semantics*, 289–294. Reidel.

Montague, Richard. 1973b. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius Moravcsik & Patrick Suppes (eds.), *Approaches to natural language: Proceedings of the 1970 Stanford workshop on grammar and semantics*, vol. 49 Synthese Library, 221–242. Dordrecht, Netherlands: Dordrecht. doi:10.1007/978-94-010-2506-5_10.

Montague, Richard. 1974a. English as a formal language. In Richmond H. Thomason (ed.), *Formal philosophy*, 188–221. New Haven: Yale University Press.

Montague, Richard. 1974b. The proper treatment of quantification in ordinary English. In Richmond H. Thomason (ed.), *Formal philosophy*, 247–270. New Haven: Yale University Press.

Montague, Richard. 1979. The proper treatment of mass terms in English. In Francis Jeffrey Pelletier (ed.), *Mass terms: Some philosophical problems*, vol. 6, 173–178. Dordrecht, Netherlands: Reidel. doi:10.1007/978-1-4020-4110-5_12.

Moulton, Keir. 2009. *Natural selection and the syntax of clausal complementation*: University of Massachusetts Amherst dissertation.

Muskens, Reinhard. 2005. Sense and the computation of reference. *Linguistics and Philosophy* 28(4). 473–504.

Ogihara, Toshiyuki. 1989. *Temporal reference in english and japanese*: University of Texas dissertation.

Oliver, Alex & Timothy Smiley. 2013. *Plural logic*. Oxford University Press.

Paris, Scott G. 1973. Comprehension of language connectives and propositional logical relationships. *Journal of Experimental Child Psychology* 16(2). 278–291. doi:10.1016/0022-0965(73)90167-7.

Parsons, Terence. 1990. *Events in the semantics of English*. Cambridge, MA: MIT Press.

Parsons, Terence. 1995. Thematic relations and arguments. *Linguistic Inquiry* 26(4). 635–662. http://www.jstor.org/stable/4178917.

Partee, Barbara. 1973. Some structural analogies between tenses and pronouns in English. *Journal of Philosophy* 70. 601–609.

Partee, Barbara. 1983/1997. Uniformity vs. versatility: The genitive, a case study, appendix to T. Janssen (1997) "Compositionality". In Johan van Benthem & Alice ter Meulen (eds.), *The handbook of logic and language*, Amsterdam: Elsevier.

Partee, Barbara. 1984. Compositionality. In Fred Landman & Frank Veltman (eds.), *Varieties of formal semantics*, 281–312. Dordrecht: Foris.

Partee, Barbara. 1989. Possible worlds in model-theoretic semantics: A linguistic perspective. In Sture Allen (ed.), *Possible worlds in humanities, arts, and sciences: Proceedings of Nobel Symposium 65*, 93–123. Berlin & New York: Walter de Gruyter.

Partee, Barbara. 2006. Lecture 1: Introduction to formal semantics and compositionality. Lecture notes for Ling 310 The Structure of Meaning.

Partee, Barbara H. 2008. Lecture 4: Noun phrases and generalized quantifiers. Lecture notes, Formal Semantics and Current Problems of Semantics, RGGU, Moscow. https://people.umass.edu/partee/RGGU_2008/RGGU084_2up.pdf.

Partee, Barbara H. & Mats Rooth. 1983. Generalized conjunction and type ambiguity. In Rainer Bäuerle, Christoph Schwarze & Arnim von Stechow (eds.), *Meaning, use and interpretation of language*, 361–383. Berlin, Germany: de Gruyter. doi:10.1515/9783110852820.361.

Partee, Barbara H., Alice ter Meulen & Robert E. Wall. 1990. *Mathematical methods in linguistics*. Dordrecht: Kluwer.

Pasternak, Robert. 2020. Compositional trace conversion. *Semantics and Pragmatics* 13(14). doi:10.3765/sp.13.14.

Pollard, Carl & Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.

von Prince, Kilu. 2019. Counterfactuality and past. *Linguistics and Philosophy* 42. 577–615.

Quine, Willard. 1951. Two dogmas of empiricism. *Philosophical Review* 60. 20–43.

Quine, Willard. 1956. Quantifiers and propositional attitudes. *Journal of Philosophy* 53. 101–111.

Quine, Willard. 1960. *Word and object*. MIT Press.

Reichenbach, Hans. 1947. *Elements of symbolic logic*. Macmillan.

Rullmann, Hotze & Lisa Matthewson. 2017. Toward a theory of modal-temporal interaction. *Language* 93(3). 589–636.

Russell, Bertrand. 1905. On denoting. *Mind* 14. 479–93.

Sassoon, Galit. 2013. A typology of multidimensional adjectives. *Journal of Semantics* 30(3). 335–380.

Scha, Remko. 1981. Distributive, collective and cumulative quantification. In Jeroen Groenendijk, Theo Janssen & Martin Stokhof (eds.), *Formal methods in the study of language*, Amsterdam, Netherlands: Mathematical Center Tracts. Reprinted in Groenendijk et al. (1984).

Schönfinkel, Moses. 1924. Über die Bausteine der mathematischen Logik. *Matematische Annalen* 92. 305–316.

Schwarz, Bernhard. 2006. Covert reciprocity and Strawson-symmetry. *Snippets* 13. 9–10.

Schwarz, Florian, Charles Clifton & Lyn Frazier. 2008. Strengthening 'or': Effects of focus and downward entailing contexts on scalar implicatures. In Jan Anderssen, Keir Moulton, Florian Schwarz & Cherlon Ussery (eds.), *Semantics and processing*, vol. 39 University of Massachusetts Occasional Papers in Linguistics, Amherst, MA: Graduate Linguistic Student Association.

Sharvy, Richard. 1980. A more general theory of definite descriptions. *The Philosophical Review* 89(4). 607–624.

Smith, Carlota S. 1997. *The parameter of aspect*. Dordrecht: Kluwer.

Stalnaker, Robert. 1978. Assertion. In *Syntax and semantics*, vol. 9, Academic Press.

Staroverov, Peter. 2007. Relational nouns and reciprocal plurality. In *Salt xvii*, 300–316.

Strawson, P. F. 1950. On referring. *Mind* 59(235). 320–344.

Tarski, Alfred. 1929. Les fondement de la géométrie des corps (foundations of the geometry of solids). *Annales de la Société Polonaise de Mathématique* 7. 29–33.

van der Auwera, Johan & Kalyanamalini Sahoo. 2020. Such similatives: a cross-linguistic reconnaissance. *Language Sciences* 81. 101217. doi:10.1016/j.langsci.2018.12.002. `https://www.sciencedirect.com/science/article/pii/S0388000117302188`. Ad hoc categorization and language: the construction of categories in discourse.

Vendler, Zeno. 1957. Verbs and times. *Philosophical Review* 66. 143–160.

Verkuyl, Henk J. 1972. *On the compositional nature of the aspects*. Dordrecht, Netherlands: Reidel.

Vikner, Carl & Per Anker Jensen. 2002. A semantic analysis of the English genitive: Interaction of lexical and formal semantics. *Studia Linguistica* 56. 191–226.

von Fintel, Kai & Irene Heim. 2011. Intensional semantics. MIT lecture notes.

von Stechow, Arnim & Atle Grønn. 2013a. Tense in adjuncts part 1: Relative clauses. *Language and Linguistics Compass* 7. 295–310.

von Stechow, Arnim & Atle Grønn. 2013b. Tense in adjuncts part 2: Temporal adverbial clauses. *Language and Linguistics Compass* 7. 311–327.

Wasow, Thomas A. 1972. *Anaphoric relations in English*: MIT dissertation.

Winter, Yoad. 2001. *Flexibility principles in Boolean semantics*. Cambridge, MA: MIT Press.

Wittgenstein, Ludwig. 1921. Logisch-Philosophische Abhandlung. *Annalen der Naturphilosophie* 14. 185–262. Also known as *Tractatus Logico-Philosophicus*.

Wunderlich, Dieter. 2012. Operations on argument structure. In Klaus von Heusinger, Claudia Maienborn & Paul Portner (eds.), *Semantics: An international handbook of natural language meaning*, vol. 3 Handbücher zur Sprach- und Kommunikationswissenschaft / Handbooks of Linguistics and Communication Science (HSK), chap. 84, 2224–2259. de Gruyter. doi:10.1515/9783110253382.2224.

Zeijlstra, Hedde. 2007. Negation in natural language: On the form and meaning of negative elements. *Language and Linguistics Compass* 1. 498–518. doi:10.1111/j.1749-818x.2007.00027.x.

Index