Invitation to Formal Semantics

(Formerly known as *Semantics Boot Camp*)

Elizabeth Coppock
&
Lucas Champollion

Draft of January 18, 2024

Preface

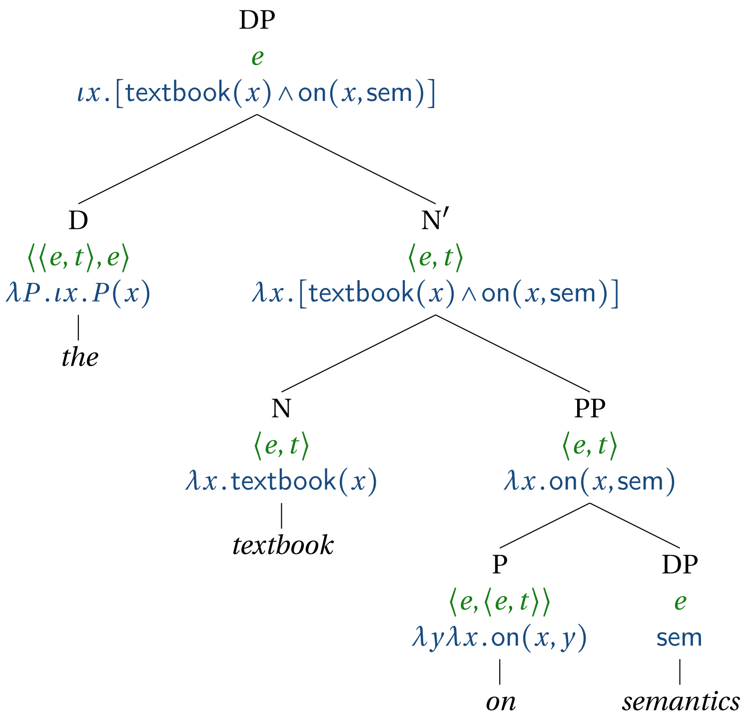
Semantics, the study of meaning, is a core subfield of linguistics, a discipline that integrates methods from the social sciences, liberal arts, and mathematics to study the nature of language. Since the 1970s, much of semantics has taken a formal turn, including techniques from mathematics such as logic 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 semantics, 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, Raef Khan, Alex Warstadt, and Nigel Flower, whose help we gratefully acknowledge.

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, Lexical Terminals, 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, and Alicia Parrish 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 David Beaver, Brian Buccola, Ivano Caponigro, Natalie Clarius, Kathryn Davidson, Alexander Stewart Davies, Thomas Grano, Magda Kaufmann, Nathan Klinedinst, Karen Lewis, Dean McHugh, Adam Przepiórkowski, Zoltán Szabó, James Walsh, Joost Zwarts, and the students of NYU Semantics I courses in 2019 and 2020.

[a full list of acknowledgements will be added in the final version]

Contents

1 Introduction

1.1 Implication

This is a book 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 Google understand language? Many might argue that it does, at least to some extent. Case in point: On July 29, 2020, we typed in "350 USD in SEK" and got back "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. But in other cases, at the time of writing, Google shows itself to be engaging rather superficially with text, without much understanding of the meaning.

How can we tell? 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 is a web page that says:

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

From this sentence, a reader would likely infer that Natalie Port-

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

man 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 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 hallmark of a system or agent that understands language—or grasps meaning—is that it can draw these kinds of inferences. In other words, a good theory of meaning should be able to explain when one sentence IMPLIES another sentence.[2]

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

[2] 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 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 that they trust the speaker).' This notion covers a wide range of subtypes of implication, including ENTAILMENTS, IMPLICATURES, and PRESUPPOSITIONS. This chapter provides a brief introduction to all three, explains how to tell them apart, and gives an overview of how, and to what extent, our theory of semantics will handle them.

1.2 Varieties of implication

1.2.1 Defining entailment

In semantics, the type of implication that lies at the core is entailment. Let us now characterize this notion. Entailment 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.

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.

The symbol ∴ is pronounced "therefore".

Each of these three sequences of sentences is an ARGUMENT, in the sense that it presents a CONCLUSION as a consequence of one or more PREMISES.[3] 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

[3] 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*.

$\not\therefore$ 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. $\not\therefore$ All animals are cats. (Conclusion)

The premise of this argument is true, but its conclusion is false. 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. $\not\therefore$ 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 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, 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

Thing in the world	Is it A?	Is it B?	Is it C?
Thing 1	yes	no	yes
Thing 2	yes	no	yes
Thing 3	yes	no	yes
Thing 4	yes	yes	no
Thing 5	yes	yes	no
Thing 6	yes	yes	no
Thing 7	yes	yes	no

Table 1.1: A particular CASE where A describes everything there is, and nothing is described both by B and by C.

Johnny, the argument might as well look like this:

(12) a. All C are A.
b. Some A are B.
c. Some C are B.

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 describes everything there is, and that nothing is described both by B and by C, like in the case described by Table 1.1.

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 Table 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', 'circumstance (of evaluation)', 'state of affairs', or 'scenario'. 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. Informally, cases can be thought of as the kinds of possibilities that Johnny considers, like the one illustrated in Table 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.
Or in slogan form: Whenever A is true, B is true too.

One important job of the semanticist is to set up a theory of when sentences in natural language stand in an entailment relation and when they do not.

As far as entailment and validity are concerned, it doesn't matter whether the premises are true in reality. These notions are is not about evaluating truth in a single case; it's 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 tell us that the argument is valid, because it has no counterexamples. In every case in which the premises are true, the conclusion is true too. As this example shows, 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. (While one might hesitate to call it a good argument, that's a different matter.)

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?

(a) Every Spaniard is female. Yo Yo Ma is a Spaniard. Therefore, Yo Yo Ma is female.

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

(c) There is no person that is not a living being. Angela Merkel is a person. Therefore, Angela Merkel is a living being.

(d) Copenhagen is either in Denmark or in the Netherlands. Copenhagen is not in the Netherlands. Therefore, Copenhagen is in Denmark.

Exercise 2. Can a valid argument have...

- false premises and a false conclusion?
- false premises and a true conclusion?
- true premises and a false conclusion?
- 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.

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 (at least assuming that *north* is a logical term and that the laws of geometry hold in all possible circumstances):

(17) a. Alaska is north of **New York**.
b. **New York** is north of Florida.
c. ∴ Alaska is north of Florida.

but the following is not:

(18) a. Florida is north of **no U.S. state**.
b. **No U.S. state** is north of Alaska.
c. $\not\therefore$ Florida is north of Alaska.

Clearly, *no U.S. state* has a very different kind of meaning from *New York*. It is a QUANTIFIER, and although quantifiers and names can occupy the same syntactic positions, they give rise to very different entailments. So the syntax is not always a reliable guide to the semantics.

Furthermore, because sentences in natural language can be AMBIGUOUS, the validity of an argument may depend on how the sentences in it are read. For example, suppose (19a) is true. Does it follow that (19b) is true?

(19) a. Today, Jane received five emails and responded to four.
b. Jane hasn't responded to an email today.

In one sense, yes, but in another sense, no. (19b) can be read either as saying that it is not the case there there is an email Jane has responded to, or that there is an email (a particular one) that she hasn't responded to. This difference is a SCOPE AMBIGUITY. In the first reading ("It is not the case that there is an email...") the negation (*n't* in *hasn't*) TAKES SCOPE over the indefinite noun phrase *an email*). In the second reading ("There is an email that she hasn't ...."), the indefinite noun phrase (*an email*) takes scope over the negation.

Another example of scope ambiguity is in the famous song

"Home on the Range":[4]

(20) ... and the skies are not cloudy all day.

Does this mean that over the course of a given day, the skies are occasionally not cloudy? Or does it mean that all day, the skies are not cloudy? It depends whether negation takes scope over the universal quantifier *all day*. The formal tools that we will develop in this book will help to elucidate the various readings that scopally ambiguous sentences can have.

When a word has multiple senses, we speak of LEXICAL AMBIGUITY. Like scopal ambiguity, this can also muddy the question of whether one sentence entails another. For example, consider the following two arguments, which are identical in syntactic structure:

(21) a. Sue and Martha are sisters.
b. ∴ Sue is Martha's sister.

versus:

(22) a. Sue and Martha are vegetarians.
b. $\not\therefore$ Sue is Martha's vegetarian.

There is at least one reading of the argument in (21) on which it is valid, but there is no reading of the argument in (22) on which it is valid. This is because (21) exhibits a kind of lexical ambiguity that is absent in (22). The validity of (21) depends on whether 'Sue and Martha are sisters' is read as 'Sue and Martha are each other's sisters', or 'Sue is a sister (to someone) and Martha is a sister (to someone)'. This ambiguity is driven by a lexical ambiguity in the noun *sister* of a kind that we will discuss later in the book. Again, we see that surface syntax is an unreliable guide to entailment.[5]

[4] We thank Jill Anderson (p.c.) for the example.

[5] Another issue for the semanticist to contend with is that many sentences are VAGUE, in the sense that there is no sharp boundary between circumstances in

Because the surface structure of sentences in natural languages is such an unreliable guide to entailment, precise and unambiguous formal languages can be a useful tool for characterizing meaning in natural language. Formal languages can help in bringing out underlying structure that is hidden in the surface form of natural language sentences. A primary aim of this book is to familiarize you with these formal methods, and empower you to develop your own variations on them.

1.2.2 Other semantic relations

Entailment is one of several different sorts of SEMANTIC RELATIONS that two sentences can stand in. Along with entailment, there is MUTUAL ENTAILMENT, or EQUIVALENCE. 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:

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

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

(25) a. John arrived before Bill left.
b. Bill left after John arrived.

There are no cases where the (a) sentences among these are false and the (b) sentences are not, or vice versa.

Sentences can also stand in various different types of opposition to each other. For instance, consider the following sentence:

---

which they are true and circumstances in which they are false. For example, a sentence like *Jane is tall* is neither clearly true nor clearly false if Jane's height is close to whatever counts as the average. As a result, it can be difficult to determine whether arguments containing vague sentences are valid. Without denying that vagueness is pervasive in natural language, we set it aside in this book.

(26) Everybody likes chocolate.

To deny this statement, you might say:

(27) Not everybody likes chocolate.

Whenever (26) is true, (27) is false, and vice versa. Together, they "divide the true and false among them" (cf. Horn 2018). In this sense, (26) and (27) stand in CONTRADICTORY OPPOSITION.

A stronger denial of (26) would come from the following:

(28) Nobody likes chocolate.

Certainly this sentence couldn't be true simultaneously with (26), but it's an extreme statement, so both could be false, as in the case where some like chocolate and some do not. Sentences (26) and (28) stand not in contradictory opposition, but rather CONTRARY opposition.

These two sorts of opposition can be defined as follows:

(29) 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 impossible for A and B to be false together.

(30) 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.

Another way of thinking about contrary opposition is that it involves two extremes, so that there is a middle ground, where neither hold.

As another example, consider the following sentence:

(31) Jo is tall.

A sentence that stands in *contradictory* opposition to (31) is *Jo* *is not tall*. If one is true, then the other must be false, and vice versa. But if we replace *tall* with its antonym, *short*, producing *Jo is short*, then the result stands in *contrary* opposition to (31). The two sentences can't both be true (in the same context), because one cannot be both tall and short at the same time (as long as we are applying consistent standards for height), but they could both be false: Jo might be neither tall nor short, just somewhere in that grey zone, the 'zone of indifference' as Sapir (1944) called it (see e.g. Kennedy & McNally 2005).

Exercise 3. For each of the following sentences, give (i) a sentence that stands in contrary opposition to it and (ii) a sentence that stands in contradictory opposition to it.

(a) My pet giraffe is young.

(b) I always drink coffee in the morning

(c) The evidence proves that he is guilty.

(d) Everyone liked it.

A famous constellation of semantic relations is the SQUARE OF OPPOSITION, shown in Figure 1.1. The square of opposition has four corners, A, I, E, and O, which come from the Latin words *affirm* and *nego*. The affirmative side is the lefthand side, where we have *Every semanticist is a philosopher* and *Some semanticist is a philosopher*, a universal and a particular (a.k.a. existential) statement, the former above the latter in the square. On the right-hand, negative side, we have *Every semanticist is **not** a philosopher* and *Some semanticist is **not** a philosopher*, again a universal and an existential. It has been a topic of discussion since Aristotle, and running through the middle ages, exactly what semantic relations hold among the corners of the square; see for example Horn

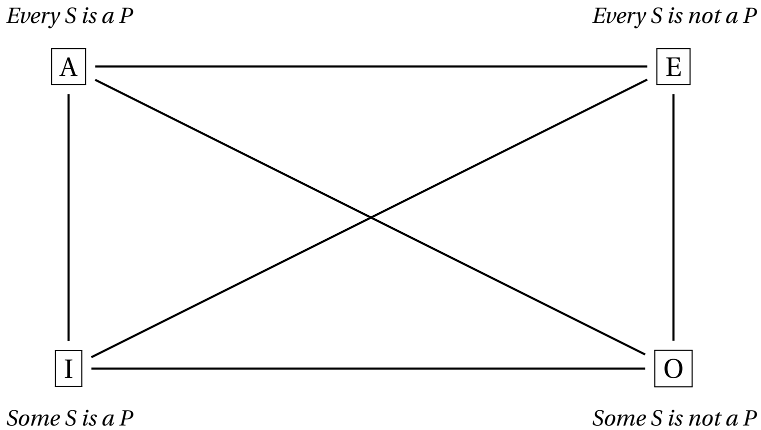


Figure 1.1: The square of opposition (excluding relations among sentences). Shorthand: *S* = *semanticist*; *P* = *philosopher*.

(2018). Nevertheless, it is fairly uncontroversial that the pairs of sentences that are connected by a diagonal line stand in contradictory opposition, while the two sentences at the top stand in contradictory opposition.

Exercise 4. Using the definitions of contradictory and contrary opposition, give examples of cases to show why the A and E corners of the square of opposition stand in contrary rather than contradictory opposition.

What all of these semantic relations have in common is that they are grounded in truth conditions. Entailment means that the set of cases that makes one sentence true is a subset of the cases that makes another sentence true; contradiction means that there is no case relative to which both sentences are true; equivalence means that any case that satisfies the one also satisfies the other; etc. Since semantics is about truth, these sorts of relations are in the purview of semantics.

The term CONTRADICTION can be used in application to a single sentence as well; a sentence that is a contradiction is one that is false in every case. Here are some examples:

(32) There is a meeting on Monday and there is no meeting on Monday.

(33) Some students are not students.

On the other hand, a TAUTOLOGY is a sentence that is always true, such as:

(34) Either there is a meeting on Monday or there is no meeting on Monday.

(35) Every student is a student.

It is also part of the job of the semanticist to explain which sentences are self-contradictory or tautologous.

1.2.3 Entailment vs. implicature

We now discuss how to distinguish entailments from another kind of implication, namely implicatures. Recall example (1), repeated here as (36):

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

This sentence 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 (36) 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 (36) 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. 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.

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. Only if the question under discussion is what she looks like does the implicature arise.

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 make up 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 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. Although scholars have debated what exactly the norms of conversation are over the years since Grice published his seminal work, we have held onto the idea that an implicature is an implication that arises crucially through reasoning about norms of conversation.

Exercise 5. 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. For example, one could say, without contradicting oneself:

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

Here, the second sentence expresses the *negation* of the implicature of the first (since the implicature was itself negative: that Natalie Portman does not speak Russian). What is to be observed about this example is that the combination of the two sentences is not contradictory; if your friend made these two claims in succession, you could not accuse her of contradicting herself. In other words, the second sentence can be used successfully to CANCEL the implicature of the first sentence that Natalie Portman does not speak Russian. Another way of putting this is that the inference is DEFEASIBLE (i.e., can be 'defeated' without contradiction).

In contrast, entailments are not defeasible. Consider:

(38) 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 (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, because the first sentence entails *Natalie Portman speaks more than two languages.*

This DEFEASIBILITY TEST is a way to to test whether the implication from a sentence *A* to sentence *B* is an entailment or an implicature. The test requires the construction of an example in which *A* is followed by a negated version of *B*, and an intuitive judgment from 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.[6] More specifically, to run the

[6]There are defeasible inferences that are not implicatures. Among these are inferences based on real-world knowledge. For example, *John smokes* loosely

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*, *in fact*, whichever fits best).

For the purposes of the defeasibility test, we define a NEGATION of a sentence as one that stands in *contradictory* opposition to it. Constructing a negation, then, is not always a matter of just adding a *not*. For example, let's consider whether (39a) entails (39b).

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

To run the defeasibility test on this example, we have to construct a negated version of (39b). In this case, rather than adding a *not*, as we just did, we can take one away: *All Republicans voted 'yes'*. Now we can ask whether *A & not-B* is SELF-CONTRADICTORY:

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

Would someone who uttered (40) be contradicting themselves? No. So *A & not-B* is not self-contradictory in this case. So the implication from (39a) to (39b) is not an entailment; it's a conversational implicature.

This *some* → not *all* pattern is a textbook example of a so-called SCALAR IMPLICATURE. Scalar implicatures arise when there is a scale of alternatives – in this case *some* and *all* arranged from weakest to strongest. In this case *some* is weaker than *all*, in the sense that it doesn't convey as much information about the way the world is. By choosing the weaker alternative rather than the stronger one, a speaker can implicate that the stronger alternative is not true, or not a good way of describing the situation. This pat-

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

tern (weak → not strong) is what constitutes a scalar implicature.

Again, we define a negation of a given sentence as one that stands in contradictory rather than contrary opposition to it. One very general strategy for creating one is to prefix *It is not the case that* to the beginning of the sentence; *It is not the case that S* is always a negated version of *S*. For example, *It is not the case that everyone likes to eat* is a negated version of *Everyone likes to eat.* Notice that this sentence is equivalent to *Not everyone likes to eat* in the following sense: when one is true, the other is true too, and when one is false, the other is false too.

An inference from A to B is defeasible if a sentence of the form *A & not-B* is not contradictory. In other words, if it's possible to assert A and deny B without contradicting oneself, then the inference is defeasible. So to run the defeasibility test, the question to ask a native speaker of the language about this type of example is 'Is this example self-contradictory?' If the answer is yes, then the *A* sentence cannot be true while the *B* sentence is false, so the implication from *A* to *B* is an entailment, rather than an implicature.

Exercise 6. Samuel Bronston was a movie producer who filed for bankruptcy in 1964 after his very expensive epic film *The Fall of the Roman Empire* failed at the box office (Solan & Tiersma, 2014, 213–221). In 1966, he was questioned under oath by his creditors regarding his overseas assets, and the exchange went as follows:

> **Q**. Do you have any bank accounts in Swiss banks, Mr. Bronston?
> **A**. No, sir.
> **Q**. Have you ever?
> **A**. The company had an account there for about six months, in Zürich.
> **Q**. Have you any nominees who have bank accounts in Swiss banks?
> **A**. No, sir.

Q. Have you ever?
A. No, sir.

It turned out that Bronston personally had had an account with International Credit Bank in Geneva. He made deposits in it and drew checks from it totalling up to $180,000 during the five years in which the company was active. He closed it just before the bankruptcy filing.

He was charged with perjury and convicted. But he appealed, and ultimately he was acquitted by the U.S. Supreme Court, who ruled that it is the responsibility of the questioner to press for further information when the respondent is 'unresponsive'.

Clearly, when he said *The company had an account there...* Bronston implied something that was not true, namely that *he himself* did *not*. But was this implication an implicature or an entailment? Argue for your answer using the defeasibility test.

Exercise 7. Consider the following pairs of sentences:

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

(42) a. Every dog barked.
b. Every dog made a noise.

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

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

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

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

In each case, the first implies the second. Are these implications entailments? Support your answers by applying the defeasibility test.

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. We leave aside the question of how to diagnose presuppositions until Chapter 8.

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 lies squarely within their intersection. However, it is fair to say that ordinary entailments lie in the domain of semantics proper, while implicatures lie in the domain of pragmatics proper. Since this is a book about semantics, implicatures will largely be left out of the discussion. We treat presuppositions in a later chapter, but our primary focus throughout the book is on ordinary entailments. In the next section, we describe our strategy for developing a theory that can account for them.

1.3 Theoretical foundations

We now describe the theoretical foundations for the family of theories developed in this book. To account for entailment relations among sentences, we will devise a system that assigns TRUTH CONDITIONS to sentences. The system will do so in a COMPOSITIONAL manner, with meanings of larger expressions built up from meanings of the parts. In these respects, this book presents quite an ordinary picture of formal semantics. The principal design feature that distinguishes this book from the otherwise quite similar textbook Heim & Kratzer 1998, is stylistic: We make use of an INDIRECT INTERPRETATION style, where natural language expressions are mapped to expressions of a representation language, which are in turn interpreted. In this respect the book is more like Dowty et al. 1981, a more traditional exposition of modern formal semantics. Let us explain all this in a bit more detail.

1.3.1 Truth-conditional semantics

1.3.1.1 What is truth-conditional semantics?

We said above that explaining entailment patterns in natural language lies in the domain of semantic theory. We also said that entailment could be characterized as follows: For any two arbitrary sentences A and B: A entails B if and only if there is no case where A is true, but B is not. In order to explain entailments, therefore, we will define an association between sentences and cases. In other words, we will need to associate sentences with their TRUTH CONDITIONS, following in the logical tradition championed by the likes of Bertrand Russell, Gottlob Frege, Alfred Tarski, Rudolf Carnap, Ludwig Wittgenstein, Donald Davidson, David Lewis, Richard Montague, and Barbara Partee, who has acted as an ambassador between philosophy and linguistics, coming from linguistics but contributing to both fields.

At some level, truth conditions are a way of characterizing the meaning of a sentence. 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 are the situations (or cases, as we have been referring to them) under which the sentence is true. They don't determine whether the sentence is in fact true, but taken together, they 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.

Exercise 8. What is *truth-conditional* semantics?

1.3.1.2 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."[7] Heim & Kratzer (1998) begin their textbook similarly: "To know the meaning of a sentence is to know its truth conditions." This opening might be taken to be making the bold suggestion 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

[7] Wittgenstein (1921) 4.024; our translation. The original German uses the word 'Satz' where we have 'sentence'. Published translations use 'proposition' instead of 'sentence', but the term 'sentence' is closer to our usage; Wittgenstein did not distinguish between sentences and propositions.

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. 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 declarative statements of fact 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.

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

In (47a), the pronoun *she* is most naturally interpreted as referring to Sue's twin. In (47b), it has to refer to Sue.

The earliest well-known example of this kind is due to Barbara

Partee (cited in Heim 1982b):

(48) 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.

DYNAMIC SEMANTICS models this as a difference in meaning, and we will illustrate how this works in Chapter 9. Until then, our system will be STATIC.

The 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 every possible circumstance—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 9. In what ways does meaning go beyond truth conditions? Discuss three.

1.3.2 Compositionality

An important goal for this book is to develop systems for natural language semantics that are COMPOSITIONAL in the sense that 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). Truth conditions will be assigned to sentences by combining together the meanings of smaller expressions (noun phrases, verb phrases, etc.).

Thanks to compositionality, along with the RECURSIVE nature of the grammar and associated semantic composition system that we will build, the parts can be combined and re-combined in infinitely many new ways. (In general, a RECURSIVE system is one in which a concept or a procedure can be defined in terms of itself, while avoiding circularity. We will illustrate how the concept of recursion applies in our case when we present the systems in detail starting in Chapter 3.) In this respect, the systems you will encounter in this book reflect the human capacity to produce and understand infinitely many new sentences (Chomsky, 1957, 1965), a core characteristic of human language.

An analogy from arithmetic might help to illustrate what it means for a semantic theory to be compositional. For the purposes of discussion, we can think of the 'meaning' of the compound expression 6*(3+2) as the number thirty. One of the parts of this expression is the digit 6, whose meaning is the number six; another part is the compound expression 3+2, whose meaning is the number five. The meaning of 6*(3+2) only depends on the meanings (six and five) of its parts (6 and 3+2) and how they are combined (in this case, by multiplication). It does not depend on anything else, such as the length or complexity of the subexpressions, or the meanings of its surroundings.

For example, the expression 3+2 has the same meaning as the expression 5, and these meanings don't depend on what surrounds these expressions. Whether we enter 6*(3+2) into a calculator or 6*5, the result is the same. In general, in a compositional system, when we substitute one part of an expression by something else that has the same meaning, the meaning of the whole expression remains the same. This is called the PRINCIPLE OF SUBSTITUTIVITY.

Exercise 10. What would it look like if the principle of substitutivity was violated? Give an example from math.

Exercise 11. What does it mean for a theory of meaning to be *compositional*?

Depending on how we make the concept of 'meaning' precise, we obtain different notions of compositionality. In this book, we will define semantic theories, sets of rules, that assign a SEMANTIC VALUE, or equivalently, DENOTATION, to each grammatical expression of the language. For instance, these rules will assign a proper noun like *Sue* a particular individual as its denotation; we will say that the noun DENOTES that individual. A common noun like *cat*, on the other hand, does not pick out any particular individual but rather could apply truthfully to any number of individuals, so its denotation is more like a *set* of individuals. (A set is just an unordered collection of objects, as we discuss in Chapter 2.) Which set? The set of actual cats currently in existence at the time of writing? We can imagine various counterfactual circumstances in which the set of cats is different. With respect to those circumstances, the word *cat* would not pick out the set of cats in the actual world currently in existence at the time of writing, but some other set. So we will say that denotations depend on the circumstance; an expression denotes whatever it denotes *with respect to a given circumstance.*

The denotation of an expression with respect to a particular circumstance is called the EXTENSION of the expression at that circumstance; the INTENSION of an expression encompasses information about its extensions at each of the various circumstances. One way of thinking about the intension of an expression is as a function (as defined in Chapter 2) that takes as input a circumstance, and returns the extension of the expression in question at that circumstance. Sometimes, the extension of a complex expression depends not only on the extensions of its parts, but also on their intensions:

(49) a. Johnny wants to find a unicorn.
b. Johnny wants to find a dragon.

Clearly these sentences don't have the same meaning (one might be true while the other one is false). By the principle of substitutivity, *unicorn* and *dragon* can't have the same meaning either. Since their extension with respect to reality is the empty set, this extension cannot be their meaning. Their intensions, however, differ, and so intensions are closer to meanings than extensions are.

A compositional system for assigning semantic values to complex expressions that allows the extension of a complex expression to depend on the intension of one or more of its parts is called INTENSIONAL. A compositional system in which the extensions of complex expressions depend only on the extensions of their parts is called EXTENSIONAL. Our compositional system will be extensional until Chapter 13, when we will incorporate tools that allow us to treat intensional phenomena.

Now, *how* is the meaning of a complex expression determined from the meanings of the parts? Frege (1891) described semantic composition of two parts metaphorically in terms of 'saturation'; one part is somehow missing something ('unsaturated'), and the other part is the missing piece (and thus 'saturates' it). A similar idea goes back at least to Aristotle's division of a sentence into subject and predicate, and shows up in modern linguistics in syntax rules such as $S \rightarrow NP\ VP$. Here is how Frege puts it:

> Statements in general [...] can be imagined to be split up into two parts; one complete in itself, and the other in need of supplementation, or "unsaturated." Thus, for example, we split up the sentence "Caesar conquered Gaul" into "Caesar" and "conquered Gaul."

> The second part is "unsaturated" — it contains an empty place; only when this place is filled up with a proper name, or with an expression that replaces a proper name, does a complete sense appear.[8]

Frege proposed to model this "saturation" using mathematical FUNCTIONS, which we will discuss in Chapter 2. We will see exactly how this works in Chapter 6.

1.3.3 Indirect interpretation

Finally, let us say a word about the style of analysis that we will adopt in this book, called INDIRECT INTERPRETATION. This is a style in which a formal language (which we refer to as a REPRESENTATION LANGUAGE) serves as an intermediary between the natural language of interest and the theory of its semantics.

To explain this more fully, let us begin with the distinction between OBJECT LANGUAGE and META-LANGUAGE. In order to give a theory of meaning for a given language (say, English), we must somehow express that theory. The language in which the theory is expressed is called the META-LANGUAGE, and the language being characterized is called the OBJECT LANGUAGE. In other words, the OBJECT LANGUAGE is the language that we are talking *about*, while the language *in which* we theorize about the object language 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 (with some mathematical notions mixed in) to theorize about English, so the object language is English, and so is the meta-language (with some mathematical notions mixed in). But if we were developing a theory of French, then French would be the object language, and we might still use English 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.

[8] Translation by Black & Geach (1961), p. 31.

Exercise 12. 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*.

As we are interested here in natural language semantics, a natural language like English or Swahili will be the language whose semantics we want to characterize. But we will use *another* language *with its own semantics* as an intermediary between this natural language and the language in which the theory is expressed. The intermediary is a formal language (an artificial language defined by clear and explicit rules), serving as what we call a REPRESENTATION LANGUAGE. 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.

Part of our task as theorists, then, is to specify the syntactic and semantic 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'.

Our theory will thus consist of two mappings:

- a mapping from expressions of a natural language to expressions of the representation language
- a mapping from representation language expressions to semantic values (described using the meta-language, which incorporates elements from set theory)

This method is known as INDIRECT INTERPRETATION, and it is the method used in Richard Montague's paper, 'The Proper Treatment of Quantification in Ordinary English' (Montague, 1974b). 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. The main advantage derives from the fact that natural language is ambiguous and vague, while our logical representation language is not. Having a non-vague, non-ambiguous representation language makes it possible to give a coherent treatment of entailment—our core phenomenon of interest—as well as related, important notions 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. Finally, and importantly, it also meshes well with the Lambda Calculator, a pedagogical software application that is integrated with this book.

Exercise 13. What is the difference between direct and indirect interpretation? Which style will be used in this book?

What to expect

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

In the previous chapter, we defined entailment as follows: *A* entails *B* if and only if there is no circumstance in which *A* is true but *B* is not. An equivalent way of characterizing entailment is in terms of SUBSET: *A* entails *B* if and only if the set of circumstances where *A* is true is a subset of the set of circumstances where *B* is true. That the premise is a subset and not a superset of the conclusion reflects the fact that premises may be more specific than conclusions: being more specific amounts to being true in fewer circumstances. Characterizing entailment is only one of the many uses for set-theoretic concepts in semantic theorizing; there are many more.

This chapter provides a brief introduction to set theory, including relations between sets like SUBSET and SUPERSET, as well as operations on sets like INTERSECTION, UNION and COMPLEMENT. We will also use sets to characterize RELATIONS and FUNCTIONS. Functions play a particularly important role in semantic theorizing, as they give us a way of making precise the idea that composition somehow involves saturation of something unsaturated.

Set theory not only lies at the foundation of the mathematical concepts used in formal semantics (and indeed of all of mathematics); it can also be applied fairly directly to some linguistic puzzles. We will introduce such a puzzle here.

2.2 Negative polarity items: the puzzle

There are certain words of English, including *any*, *ever*, *yet*, and *anymore*, which can be used in negative sentences but not all positive sentences (at least in standard varieties of English):

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

The italicized words in these examples are called NEGATIVE POLARITY ITEMS (NPIs). The contrast between (1a) and (1b) shows that negative polarity items are licensed in the presence of negation. Specifically, they are licensed in ENVIRONMENTS containing negation. An environment is the part of a sentence that surrounds a given constituent (in this case, the constituent that contains the NPI).

It's not just environments containing negation where NPIs can be found. Here is a sampling of the data (Ladusaw, 1980).

(2) $\left\{\begin{array}{l}\text{No one}\\ \text{At most three people}\\ \text{*Someone}\\ \text{*At least three people}\\ \text{*Many students}\end{array}\right\}$ who had *ever* read *anything* about phrenology attended *any of the lectures*.

(3) I $\left\{\begin{array}{l}\text{never}\\ \text{rarely}\\ \text{seldom}\\ \text{*usually}\\ \text{*always}\\ \text{*sometimes}\end{array}\right\}$ *ever* eat *anything* for breakfast *anymore*.

(4) a. Susan finished her homework $\left\{\begin{array}{l}\text{without}\\ \text{*with}\end{array}\right\}$ *any* help.
b. Susan voted $\left\{\begin{array}{l}\text{against}\\ \text{*for}\end{array}\right\}$ *ever* approving *any* of the pro-

posals.

(5) John will replace the money $\left\{\begin{array}{l}\text{ before}\\ \text{ if}\\ \text{*after}\\ \text{*when}\end{array}\right\}$ *anyone ever* misses it.

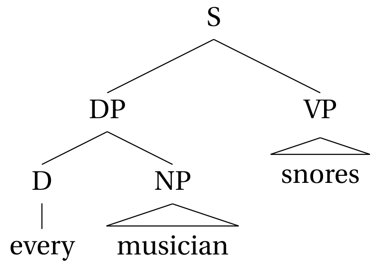
(6) It's $\left\{\begin{array}{l}\text{ hard}\\ \text{ difficult}\\ \text{*easy}\\ \text{*possible}\end{array}\right\}$ to find *anyone* who has *ever* read *anything much* about phrenology.

(7) John $\left\{\begin{array}{l}\text{ doubted}\\ \text{ denied}\\ \text{*believed}\\ \text{*hoped}\end{array}\right\}$ that *anyone* would *ever* discover that the money was missing.

(8) It $\left\{\begin{array}{l}\text{ is unlikely}\\ \text{ is doubtful}\\ \text{ amazed John}\\ \text{*is likely}\\ \text{*is certain}\\ \text{ is surprising}\\ \text{*is unsurprising}\end{array}\right\}$ that *anyone* could *ever* discover that the money was missing.

So, along with negation, there are words like *hard* and *doubt* and *unlikely* which license negative polarity items. What could these words have in common?

The issue is made a bit more complex by the fact that words differ as to *where* they license negative polarity items. Words like *every*, *some*, and *no* belong to the syntactic category of DETERMINERS (as opposed to nouns, verbs, adjectives, etc.), and these determiners differ as to where they license NPIs. For the purposes of discussion, let us assume the following syntactic structure for sentences like *Every musician snores*:



A determiner like *every* is of category D (for determiner), and we assume that it combines with an NP (for 'noun phrase', used in this book for a phrase headed by a noun but excluding any determiners) to form a DP (for 'determiner phrase', used in this book for a phrase headed by a determiner). We are thus following the 'DP hypothesis' (Abney, 1987), according to which a phrase like *every musician* is headed by the determiner *every* rather than the noun *musician*, as opposed to the 'NP hypothesis', according to which phrases like *every musician* are headed by nouns, hence NPs. The term "noun phrase" is sometimes used to refer to things like *every musician*, regardless of whether they are analyzed as NPs or DPs. In this book, we sometimes follow this practice in cases where it will not lead to confusion, but reserve the term NP for things like *musician*, and DP for things like *every musician*. The DP and the VP together form a complete sentence, of category S. (The triangles in the trees indicate that there is additional structure that is not shown in full detail—in this case, the links from NP to N and VP to V.)

No licenses NPIs throughout the sentence, in both the NP and the VP:

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

And *some* fails to license NPIs both in the NP and in the VP:

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

But *every* licenses NPIs only in the NP, *not* in the VP:

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

This shows that the ability to license negative polarity items is not a simple yes/no matter for each lexical item.

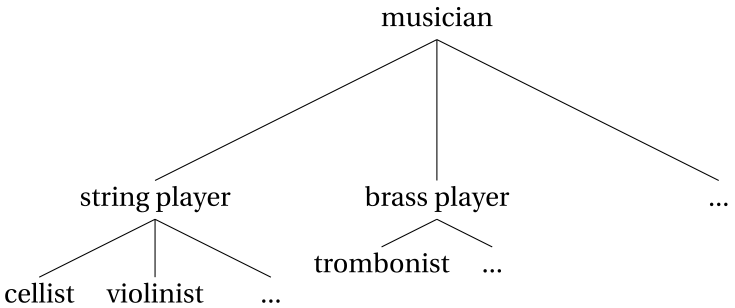
Building on work by Fauconnier (1975), Ladusaw (1980) illustrated a correlation between NPI licensing and "direction of entailment". A simple, positive sentence containing the word *cellist* will typically entail the corresponding sentence containing the word *musician*, as shown by the validity of the following argument:[1]

(12) Mary is a cellist.
∴ Mary is a musician.

The "direction of entailment" can be either DOWNWARD or UPWARD. To understand the idea behind these labels, let us arrange terms like *cellist* and *musician* visually in a TAXONOMIC HIERARCHY (an arrangement of categories specifying which categories

[1] As mentioned in Chapter 1, there are a number of different precise notions of consequence. Among them are 'logical consequence' and 'necessary consequence'. The argument in (12) is 'necessarily valid' because it is impossible for the conclusion to be false when the premise is true, assuming that being a musician is an intrinsic part of what it means to be a cellist. But it is not 'logically valid', because it is not valid solely in virtue of its form. Similar remarks apply to subsequent examples in this chapter.

are sub-categories of others, sometimes used in biology to capture the organization of flora or fauna into categories) with more specific concepts below more general concepts.



In terms of this visual representation, the inference in (12) proceeds from lower (more specific) to higher (more general), hence "upwards". An entailment by a sentence of the form [... *A* ...] to a sentence of the form [... *B* ...] where *A* is more specific than *B* can thus be labelled an UPWARD ENTAILMENT. The validity of the following argument illustrates another upward entailment:

(13) Some cellists snore.
∴ Some musicians snore.

But negation and the determiner *no* reverse the entailment pattern; witness the validity of the following arguments:

(14) Mary isn't a musician.
∴ Mary isn't a cellist.

(15) No musicians snore.
∴ No cellists snore.

The entailments here are called, as you might guess, DOWNWARD ENTAILMENTS, because they go from more general (higher) to more specific (lower).

We can also consider direction of entailment between sentences varying only in the verb phrase (VP). If we think of *snores* and *snores loudly* as being arranged in the same kind of taxonomic hierarchy, where the more general terms are higher and the more specific terms are lower, we see that *snores loudly* is below *snores*, because every case of snoring loudly is a case of snoring.

snores

snores loudly   snores softly

In simple sentences like the following, replacing a specific VP with a more general one yields a valid argument, so the VP is in an UPWARD-ENTAILING ENVIRONMENT.

(16) Ed snores loudly.
∴ Ed snores. (upward)

But negation reverses the direction of entailment, so that the VP is then in a DOWNWARD-ENTAILING ENVIRONMENT:

(17) Ed doesn't snore.
∴ Ed doesn't snore loudly. (downward)

It turns out that there is a correlation between NPI-licensing and downward entailment: NPIs are licensed in downward-entailing environments. Compare the following examples to the NPI data for *no*, *some* and *every* above.

(18) a. No musician snores.
∴ No cellist snores. (downward)
b. No musician snores.
∴ No musician snores loudly. (downward)

(19) a. Some cellists snore.
∴ Some musicians snore. (upward)
b. Some musician snores loudly.
∴ Some musician snores. (upward)

(20) a. Every musician snores.
∴ Every cellist snores. (downward)
b. Every musician snores loudly.
∴ Every musician snores. (upward)

Observe that there is an exact match between the environments that are downward-entailing and those in which negative polarity items are licensed. This exact match is in accordance with the FAUCONNIER-LADUSAW GENERALIZATION: *An expression licenses negative polarity items wherever it licenses downward entailments.*

What we have said so far about what it means for one expression to be "downward" of another one is that it is more "specific". But as Farkas (2002, 213) writes, "the notion of specificity in linguistics is notoriously non-specific." We can state this idea more precisely by making use of the certain technical vocabulary, in particular, the concepts of SET and SUBSET. A SET is an abstract collection of distinct objects, which are called the MEMBERS or ELEMENTS of that set. One set is a SUBSET of another set if and only if every member of the first is a member of the second (or in other words: there is nothing in the first that isn't also in the second, although there may be elements of the second that are not in the first).

Let us assume that words like *cellist* and *musician* denote sets, such as the set of cellists and of musicians. While the set of cellists in one circumstance may differ from the set of cellists in another, it does not matter which circumstance we pick, since regardless of our choice, every cellist is a musician; so, every member of the set denoted by *cellist* is bound to be a member of the set denoted by *musician*. Hence the denotation of *cellist* is always (in all circumstances) a subset of the denotation of *musician*. Assuming that verb phrases like *snores* and *snores loudly* denote sets as well, we again have a subset relation: every member of the set denoted by *snores loudly* (i.e. every loud snorer) is a member of the set denoted by *snores* (i.e. is a snorer).

In the next section, we turn to a more technical presentation of sets and related notions. Not only will these notions help to elucidate what we have said so far, they will also allow us to characterize the meaning of *no, some* and *every* in a way that helps to explain why they are downward-entailing where they are. These notions will also lay the foundations for the rest of this book, as our language for describing denotations (our META-LANGUAGE) builds on concepts and notational devices related to sets.

2.3 Sets

As mentioned above, a SET is an abstract collection of distinct objects, which are called the MEMBERS or ELEMENTS of that set.[2] 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." 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 an element of the set containing 2, 7 and 10.'

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

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

[2] 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 exactly the same set as this set:

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

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:

$$\{3\}$$

Sets can be very big or very small. Here is another example of a set:

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

The ELLIPSIS NOTATION $(\dots)$ signals that the list of elements continues according to the pattern. So this set is infinite; it contains all positive even numbers. But 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. The set with no elements at all is called the EMPTY SET, written either like this:

$$\{\}$$

or like this:

$$\varnothing$$

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:

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

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

The members of a set can be all sorts of things. A set can, for example, 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.

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

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). 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 ...

The only kinds of things that cannot be members of a given set are certain other sets. 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.

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.

Two sets are EQUAL if and only if they have the same members. Thus it does not matter which order we use if we list the members of a set, and which notation we use to begin with. For example, the expressions $\{1,2,3\}$, $\{1,3,2\}$, and $\{3,2,1\}$ all denote the same set.

It is important to distinguish between *elements* and *subsets*. The set $\{2,3\}$ is not an *element*, but rather a *subset* of the set $\{2,3,4\}$. In general, as we have said earlier, a set A is a SUBSET of a set B if and only if every member of A (if any) is a member of B. Put more formally:

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

This formulation uses several pieces of mathematical jargon. The word "iff" is shorthand for "if, and only if". The symbol $\subseteq$ is pronounced "is a subset of". The symbol $\in$ is pronounced "is an element of" or "is a member of" or "is contained in".

Let a, b, and c stand for three arbitrary distinct things, such as your three favorite moments, the Three Musketeers, or three particular sets of numbers. Here are some true statements:

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

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

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

Things get slightly trickier to think about when the elements of the sets involved are themselves sets. Here is another true statement:

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

(The slash across the $\subseteq$ symbol negates it, so $\nsubseteq$ can be read 'is not a subset of'.) The reason that $\{a,\{b\}\}$ is *not* a subset of $\{a,b,c\}$ is that the first has a member that is not a member of the second, namely $\{b\}$. It is tempting to think that $\{a,\{b\}\}$ contains b as an element but this is not correct. The set $\{a,\{b\}\}$ has *exactly two* elements, namely: a and $\{b\}$. The set $\{b\}$, called the SINGLETON of

b, is not the same thing as b. One is a set and the other might not be. (Whether b is a set is open at this point, since we have made no assumptions about what b is.) Likewise, the set $\{\{b\}\}$ is not the same thing as the set $\{b\}$, and so on. This example can also help clarify the difference between membership and subsethood. A member or element ($\in$) of a set is not a subset ($\subseteq$) of that set. Unlike subsethood, membership is not transitive. For example, the set $\{\{b\}\}$ has a single element, $\{b\}$, which in turn has a single element, b. In this example, b is an element of $\{b\}$ but not of $\{\{b\}\}$, and $\{b\}$ is an element, but not a subset, of $\{\{b\}\}$. As for b itself, it is not a subset of anything (assuming it is not itself a set).

The following is a true statement:

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

Every element of $\{a, \{b\}\}$ is an element of $\{a, \{b\}, c\}$, as we can see by observing that the following two statements hold:

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

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

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. (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.)

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 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:

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

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

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:

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

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 a member of A.

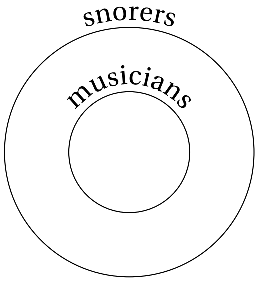
$$A \supseteq B \text{ iff for all } x\text{: if } x \in B \text{ then } x \in A.$$

Two sets are identical whenever they have the same members. This means we have $A = B$ if and only if both $A \subseteq B$ and $A \supseteq B$.

And 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.

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

The word *every* can be thought of as a relation between two sets X and Y which holds if X is a subset of Y, i.e., if every member of X is a member of Y. The sentence *every musician snores,* for instance, expresses that every member of the set of musicians is a member of the set of people who snore. This type of scenario can be depicted as follows:



Subset and superset are RELATIONS BETWEEN SETS, which 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.

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.

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

For example:

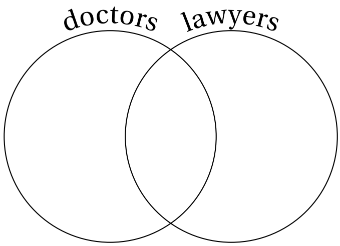
$$\{a, b, c\} \cap \{b, c, d\} = \{b, c\}$$

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

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

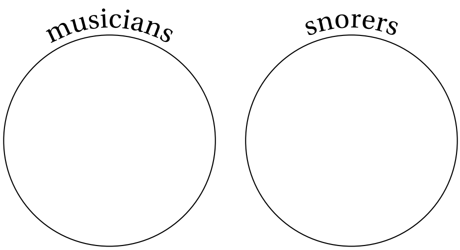
$$\{a, b\} \cap \{a, b\} = \{a, b\}$$

Intersection is very useful in natural language semantics. It can be used as the basis for a semantics of *and*. For example, if someone tells you that John is a lawyer *and* a doctor, then you know that John is in the *intersection* between the set of lawyers and the set of doctors. If the circle on the left in the following diagram represents the set of doctors, and the circle on the right represents the set of lawyers, then John is located somewhere in the area where the two circles overlap, as long as he is both a doctor and a lawyer. (This way of representing the relations among sets is called an EULER DIAGRAM.)

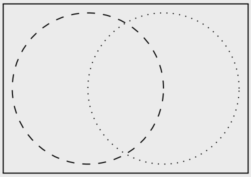
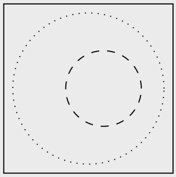


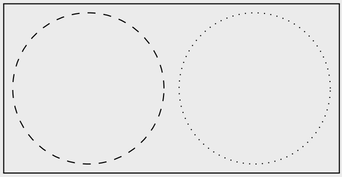
The English determiner *some* can be thought of in terms of intersection as well, as a relation between two sets X and Y which holds iff there is some member of X which is also a member of Y, i.e., iff the intersection between X and Y is non-empty. For instance, *some musician snores* is true iff there is some individual which is both a musician and a snorer.

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, iff the *intersection is empty*. So *no musician snores* holds iff there is no individual who is both a musician and a snorer. In that case, the two sets are DISJOINT, like so:



Exercise 2. Here are three Euler diagrams:

(a)  (b) 

(c) 

And here are three statements in set-theoretic language:

1. $A \cap B = \varnothing$
2. $A \cap B \neq \varnothing$
3. $A \subseteq B$

For each of the Euler diagrams, say (i) which of the three set-theoretic statements it matches, and (ii) which of the following three determiners it best represents: *some*, *every*, or *no*. (The dashed line represents A and the dotted line represents B.)

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 <u>or</u> in B (or both).

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

For example:

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

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

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

As you can guess, union can be used to give a semantics for *or*. If someone tells you that John is a lawyer or a doctor, then you know that John is in the union of the set of lawyers and the set of doctors. (You might normally assume that he is not in the intersection of doctors and lawyers though – that he is either a doctor or a lawyer, but not both. This is called an *exclusive* interpretation for *or*, and we will get to that later on.)

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

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 of entities. The complement of A can be written $\bar{A}$.[3]

Exercises on sets

The following exercises are taken from Partee et al. 1990, *Mathematical Methods in Linguistics*.

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

Exercise 4. Given the following sets:

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

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 5. Consider the following sets:

$$\begin{array}{ll} S_1 = \{\{\varnothing\},\{H\},H\} & S_6 = \varnothing \\ S_2 = H & S_7 = \{\varnothing\} \\ S_3 = \{H\} & S_8 = \{\{\varnothing\}\} \\ S_4 = \{\{H\}\} & S_9 = \{\varnothing,\{\varnothing\}\} \\ S_5 = \{\{H\},H\} & \end{array}$$

(a) Of the sets $S_1 - S_9$, which are members of S_1?

(b) Which are subsets of S_1?

(c) Which are members of S_9?

(d) Which are subsets of S_9?

(e) Which are members of S_4?

(f) Which are subsets of S_4?

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

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

list the members of each of the following:

(a) $B \cup C$

(b) $A \cup B$

(c) $D \cup E$

(d) $B \cup G$

(e) $D \cup F$

(f) $A \cap B$

(g) $A \cap E$

(h) $C \cap D$

(i) $B \cap F$

(j) $C \cap E$

(k) $B \cap G$

(l) $A - B$

(m) $B - A$

(n) $C - D$

(o) $E - F$

(p) $F - A$

(q) $G - B$

Exercise 7. Let $A = \{a, b, c\}$, $B = \{c, d\}$, $C = \{d, e, f\}$. Calculate the following:

(a) $A \cup B$

(b) $A \cap B$

(c) $A \cup (B \cap C)$

(d) $C \cup A$

(e) $B \cup \varnothing$

(f) $A \cap (B \cap C)$

(g) $A - B$

(h) Is a a member of $\{A, B\}$?

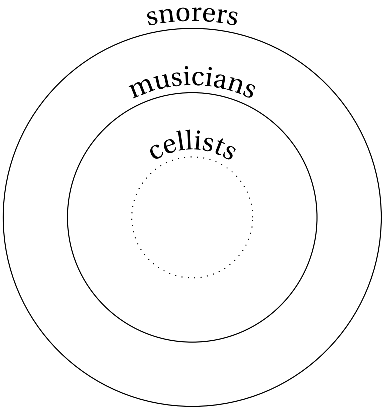
(i) Is a a member of $A \cup B$?

2.4 Negative polarity items revisited

We have characterized the truth conditions of the determiners *no*, *some* and *every* as follows:

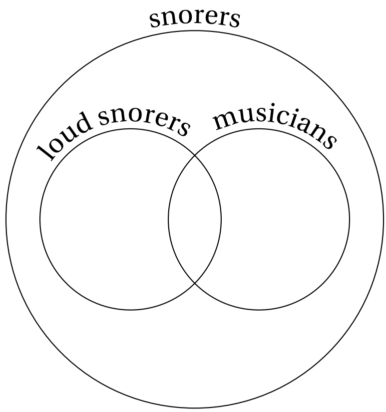
- *Every* X Y is true if and only if X is a subset of Y.
- *Some* X Y is true if and only if there is a non-empty intersection between X and Y.
- *No* X Y is true if and only if X and Y have an empty intersection.

Given this, consider what happens when we consider a subset X' of X (e.g., if X is the set of musicians, take X' to be the set of cellists). *Every* X Y is true if and only if X is a subset of Y. If that is true, then any subset X' of X will also be a subset of Y. This can be visualized as in the following Euler diagram. Assume it is true that all musicians snore. Then the set of musicians is a subset of the set of snorers. And since all cellists are musicians, the set of cellists is a subset of the set of musicians:



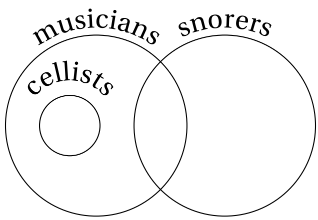
So, from *Every musician snores* it follows that *Every cellist snores*. In this particular example, we have taken X to be the set of musicians, X' the set of cellists, and Y the set of snorers. In general, *every* X Y entails *every* X' Y whenever X' is a subset of X. We say that *every* is LEFT DOWNWARD MONOTONE ("left" because it has to do with the expression on the left, X, rather than the expression on the right, Y.) In general, a determiner δ is left downward monotone iff $\delta X Y$ entails $\delta X' Y$ for all X' that are subsets of X.

A determiner δ is RIGHT DOWNWARD MONOTONE iff $\delta X Y$ entails $\delta X Y'$ for any Y' that is a subset of Y. Let us consider whether *every* is right downward monotone. Suppose that *every* X Y is true. Then X is a subset of Y. Now we will take a subset of Y, Y'. Are we guaranteed that X is a subset of Y'? No! Here is a counterexample:



Here, X (musicians) is not a subset of Y' (loud snorers). Or think about it this way: From *every musician snores* it doesn't follow that every musician snores loudly. So *every* is not right downward monotone.

Now let us consider *some*. With *some*, we are not guaranteed that a true sentence will remain true when we replace X with a subset X'. *Some* X Y is true iff the intersection of X and Y contains at least one member. If we take a subset X' of X, then we might end up with a set that has no members in common with Y, like this:



So, for example, suppose that *Some musician snores* is true. From this it does not follow that *Some cellist snores* is true, because it could be the case that none of the musicians who snore are cellists. So *some* is not left downward monotone. By analogous reasoning, it isn't right downward monotone either.

Exercise 8. Is *no* left downward monotone? Is it right downward monotone? Explain. In a sentence of the form *No* X Y, where are negative polarity items licensed (see above)? So, does the Fauconnier-Ladusaw generalization hold up for *no*? Explain.

Exercise 9. Consider the following data:

(21) At most five [of the cities I have **ever** visited] [have decent bike infrastructure].

(22) At most five [of the cities I have visited] [have **any** decent bike infrastructure **at all**].

(23) *At least five [of the cities I have **ever** visited] [have decent bike infrastructure].

(24) *At least five [of the cities I have visited] [have **any** decent bike infrastructure **at all**].

This shows that *at most five* licenses NPIs in the NP it forms a syntactic unit with as well as the VP, and *at least five* licenses negative polarity items in neither position. Let us consider whether the distribution of negative polarity items with these quantifiers fits the Fauconnier-Ladusaw generalization about downward entailment (that NPIs are licensed in downward-entailing environments).

In particular, consider whether *at most five* and *at least five* produce downward-entailing environments both in NP, and in the VP. You'll need to construct four pairs of examples, one pair for each of the environments under consideration.

Note: Your examples should **not** contain NPIs; your goal is just to determine whether the environment is downward-entailing.

Based on your observations, does the Fauconnier-Ladusaw generalization hold up for *at least* and *at most*? Explain your reasoning for your answer.

Exercise 10. For each of the examples in (2), (3), and (4b) on page 48, check whether the Fauconnier-Ladusaw generalization holds up. Are downward entailments licensed in exactly the places where NPIs are licensed? (The examples that you need to construct in order to test this should not contain NPIs; they can be examples like the ones in (18b), (20b) and (19b).)

What we have seen so far is that the Fauconnier-Ladusaw generalization works quite well as a way of characterizing the environments where negative polarity items are licenced. But it is not perfect. For example, consider the fact that *only* licenses negative polarity items in the VP in sentences like the following:

(25) Only Sandy did *any* work.

The verb phrase is not a downward-entailing environment, as shown by the fact that (26a) does not entail (26b).

(26) a. Only Sandy did work.
b. Only Sandy did gardening.

Suppose that the only kind of work that was done was food preparation and clean-up; nobody did any gardening. Then (26a) could be true even though (26b) is not true; Sandy didn't garden. This particular issue can be resolved by replacing 'downward entailment' with what von Fintel (1999) calls STRAWSON DOWNWARD-ENTAILMENT. An environment is Strawson downward-entailing if it is downward-entailing *under the assumption that all of the presuppositions of both sentences are true*. For instance, (26b) presupposes that Sandy did gardening, and (26a) presupposes that Sandy did work. Under these assumptions, (26a) does entail (26b), so the verb phrase is a Strawson downward-entailing environment.

Another type of example that is challenging for the Fauconnier-Ladusaw generalization is questions, like:

(27) Did you have *any* problems?

It is not entirely clear what it means for one question to entail another. On the basis of this and other data, some authors, including Zwarts (1995) and Giannakidou (1999), have offered a theory of negative polarity item licensing based on a notion called 'veridicality', which we will not go into here. For further reading on negative polarity items, we recommend the overview by Penka (2016) as a place to start.

2.5 Relations and functions

The denotations of common nouns like *cellist* and intransitive verbs like *snores* are often thought of as sets (the set of cellists, the set of individuals who snore, etc.). Transitive verbs like *love*, *admire*, and *respect* are sometimes thought of as denoting RELATIONS between two individuals. Relations can be modelled mathematically using pairs of elements that stand in a specified order to each other, i.e. ORDERED PAIRS.

2.5.1 Ordered pairs

As stated above (p. 55), 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 11. 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) $\{3,\{3,4\}\} = \{3,\{4,3\}\}$

2.5.2 Relations

As mentioned above, the semantics of transitive verbs like *love*, *admire*, and *respect* is sometimes modeled 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. Not all elements of A and B need necessarily be involved in the relation. The DOMAIN is the set of those entities in A that occur as a first member of some pair, and the RANGE is the set of those entities in B that occur as a second member of some pair. The union of the domain of a relation with its range is called the FIELD of a relation. The sets A and B themselves are called the DOMAIN OF DEFINITION and the CODOMAIN of the relation. Formally, a binary relation over A and B is a (proper or non-proper) subset of the Cartesian product $A \times B$.

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 'standing next to' relation is symmetric; hence the following argument is valid:

(28) Paul is standing next to George.
$\therefore$ George is standing next to Paul.

The 'admires' relation is not, though.

(29) 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:

(30) 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 a, b, and c, if $\langle a,b\rangle$ and $\langle b,c\rangle$ are in the relation, then $\langle a,c\rangle$ is also in the relation. (This notion TRANSITIVE should not be confused with the notion of a transitive verb.) Another example of a transitive relation is 'before': If a is before b, and b is before c, then a is before c.

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. 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 equivalance relation determines the partition. This notion comes up in the analysis of questions, although we will not touch on that in this book.

Exercise 12. One of the following arguments is valid, and the other is not.

(31) The singer is the drummer's brother.
$\therefore$ The drummer is the singer's brother.

(32) The singer is the drummer's sibling.
$\therefore$ The drummer is the singer's sibling.

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

(33) 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.

(34) 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 14. 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.

2.5.3 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 an INPUT (e.g. a specification of which item you would like to buy), and returns an OUTPUT (e.g. a particular bag of chocolate-covered raisins). (Let us set aside the fact that vending machines typically also require money; this constitutes an additional input.) 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 (the political leaders of Germany and New Zealand, respectively, at the time of writing) 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 2.1, the relations depicted are *not* functions. In Figure 2.2, the relations depicted *are* functions.

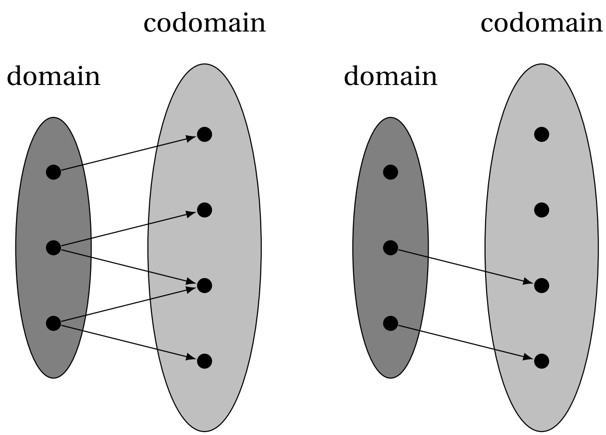


Figure 2.1: Two non-functions

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

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

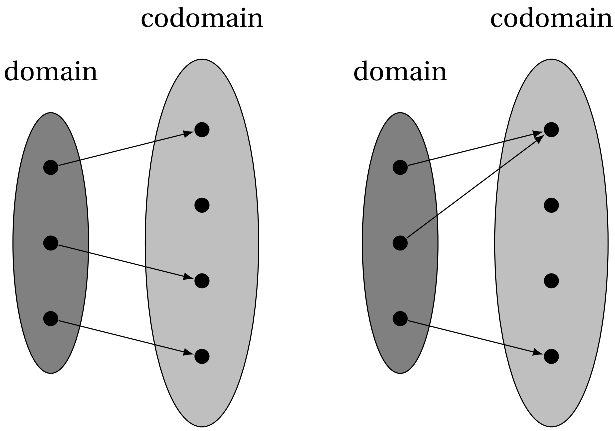


Figure 2.2: Two functions

or using large brackets like this:

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

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 15. 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 returns 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.)

The denotations (relative to a particular circumstance) 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 16. 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 17. 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 $\mathsf{male}(\text{Björn})$?

(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*?

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 in a formula of propositional logic. Here is an example of an interpretation function for L_{Prop}.

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

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* (often abbreviated *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]
∴ [not p] and [not q]

as is its converse,

(9) [not p] and [not q]
∴ 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':

(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]
∴ [not p] or [not q]

Its converse is valid as well:

(12) [not p] or [not q]
∴ 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 ~) 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 the connectives. Connectives with this property are called TRUTH-FUNCTIONAL. In Chapters 12 and 13, 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* true, *B* is true as well. Similarly, when *A* and *B* have the same truth values in every case, we say they are EQUIVALENT. 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 *there is no circumstance in which A is true but B is not.* 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. Recall from Chapter 1 that we sketched two ways of viewing entailment, the first in terms of logical consequence and the second in terms of necessary consequence. To recap, 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 13, 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 modeled 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 square brackets that go with binary connectives are always there according to the official rules of the syntax. 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

[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.

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. Above, we suggested that the inclusive reading is what *or* really denotes, and that the exclusive reading arises via a conversational implicature in certain contexts. One argument came from the fact that if one says *Kim did not invite Pat or Sandy*, it follows that Kim did not invite Pat and also did not invite Sandy.

Spell out this argument. To this end, write out truth tables for $\neg[p \vee q]$ and $\neg[p \text{ XOR } q]$ (for the definition of XOR as exclusive *or*, see above). Where do they differ? Which analysis captures the intuition that *Kim did not invite Pat or Sandy* entails that Kim did not invite Pat and also did not invite Sandy? Explain. You may assume that conversational implicatures of the kind that would be involved here ('scalar implicatures') typically disappear under negation.

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 $\to$ (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 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.

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

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. Let us consider the question of whether **Modus Tollens** ($[p \to q]$; $\neg q$; therefore $\neg p$) turns out to be a valid argument form. Start by filling in this truth table:

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

To determine whether the argument is predicted to be valid, we need to determine whether *the conclusion of the argument is true in every case where all of the premises are true.* So first, we need to determine in what cases all of the premises are true. There are two premises in the Modus Tollens argument: $[p \to q]$ and $\neg q$. The first step is to identify the row(s) in which both of these premises are true. The next step is to consider whether the conclusion of the argument ($\neg p$) is true in every such row.

With all this in mind, explain in your own words how we can see from the truth table above that Modus Tollens is valid.

Exercise 13. Recall that Denying the Antecedent has the form:

Premise 1:	$[p \to q]$
Premise 2:	$\neg p$
Conclusion:	$\neg q$

Using a truth table, explain in your own words why the argument is or is not valid, sticking closely to the truth table. (Which are the rows where all of the premises are true? Is the conclusion true in those rows?)

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.

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

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

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

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

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

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

(f) $[p \rightarrow p]$; $[p \vee \neg p]$
(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 $[p \rightarrow q]$ and $[p \wedge \neg q]$.

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

Exercise 15. 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.

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 L_{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.

4 Predicate logic

4.1 From propositional logic to predicate logic

In natural languages, sentences (clauses) may or may not consist of other sentences (clauses). For example, the English sentence *Abelard is happy and Eloise is sad* contains two sub-sentences (sub-clauses), *Abelard is happy*, and *Eloise is sad*.[1] These latter two sentences do not contain any other sentences and in that sense they can be said to be ATOMIC. Formulas are like this too: an ATOMIC FORMULA contains no other formulas, while a COMPLEX FORMULA contains other formulas. In this respect, propositional logic mirrors natural language accurately.

But in many respects, propositional logic is much simpler than natural language. While we might represent *Abelard is happy and Eloise is sad* as $[p \wedge q]$, and its first conjunct *Abelard is happy* as p, we cannot break it down further. There is nothing in propositional logic that corresponds to *Abelard* or to *happy*. There are many valid arguments that we would like to be able to capture that depend on our ability to break the units into smaller parts; for

[1] Peter Abelard was a philosopher and theologian in 12th century Paris, arguably the greatest logician of the Middle Ages and an important thinker on reason and religion. His affair with Eloise, already renowned for her knowledge of Latin, Greek and Hebrew when she arrived in Paris as a young woman, led to their secret marriage and, tragically, to his castration. At that point, Abelard became a monk, and Eloise a nun (and eventually a prioress). Their subsequent correspondence is among the most moving and personal documents of the 12th century.

example, *Abelard is happy* entails *Someone is happy*. We've seen similar arguments at the outset of the previous chapter, and we'll see plenty more as we proceed.

To gain a better tool for representing natural language, we will now "split the atom". This is the point where we go beyond the resources of propositional logic and move into PREDICATE LOGIC. The propositional letters and connectives of propositional logic all carry over to predicate logic. But in predicate logic, atomic formulas may be built up of several BASIC EXPRESSIONS—symbols that have no internal structure even in predicate logic: names like a (for Abelard) and e (for Eloise), predicate symbols like happy and loves, and function symbols like ageInYearsOf or motherOf. Constrained by the syntactic rules that we will define for the language, these basic expressions may be put together in various ways to form atomic formulas. Predicate logic also has VARIABLES and QUANTIFIERS, which we will delay until Section 4.2.

4.1.1 Individual constants

Predicate logic is a formal language that allows us to reason about a given DOMAIN of entities. Individual objects are named by INDIVIDUAL CONSTANTS, also known as NAMES. In this book, we adopt the convention that individual constants start with a lowercase letter. In general, constants (including individual constants, predicate symbols, and function symbols) may contain any sequence of letters and numbers and underscores, but no spaces. For example,

sam_smith

is a valid individual constant, but

S

is not, nor is

sam smith.

Individual constants make up one kind of TERM in the logic. A term is an expression that picks out an individual object in the domain (like a proper name such as *Sam* or a definite description such as *the sun* in natural language). Later, we will introduce variables, which are another type of term.

Recall that in propositional logic, expressions are interpreted relative to an interpretation function I, which maps propositional letters to truth values. In predicate logic, this interpretation function is given a more complex set of tasks, because it has to provide denotations for all of the basic expressions of the language. One of its jobs is to map individual constants to individuals (that is to specify which individuals the names REFER to). It is customary to write the set of individuals that are available for this purpose as D. A pair $\langle D, I \rangle$, where D is a nonempty set and I is an interpretation function, is called a MODEL for predicate logic, and we will use M to refer to an arbitrary model of this kind. The set of individuals D is called the UNIVERSE OF DISCOURSE or the DOMAIN of the model in question. (Subscripts can be placed on M, D, and I when different models and their components need to be distinguished. For example, suppose $M_1 = \langle D_1, I_1 \rangle$, and $M_2 = \langle D_1, I_2 \rangle$. This means that M_1 and M_2 are distinct models that share a domain.)[2]

To illustrate, we will define a language L_0 and interpret it in models whose domain consists of the four members of the Swedish pop band ABBA, whose names are *<u>A</u>gnetha*, *<u>B</u>jörn*, *<u>B</u>enny*, and *<u>A</u>nni-Frid* (better known by her nickname Frida). Our language L_0 should contain expressions that refer to these individuals. Let us assume that expressions in L_0 may contain the following individual constants: $\mathsf{a}, \mathsf{b}, \mathsf{e}, \mathsf{f}$. Let us also assume that our interpretation function maps these constants to the four band members in the order we have mentioned them. For this purpose, we will define the set D_0 as $\{$ Agnetha, Bjorn, Benny, Frida $\}$, the set of ABBA

[2]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. Thus, the domain of a model is a subset of the codomain (and not of the domain) of the model's interpretation function.

band members. We will also define an interpretation function I_0 as follows:

(1) a. $I_0(\mathsf{a}) = \text{Agnetha}$
b. $I_0(\mathsf{b}) = \text{Björn}$
c. $I_0(\mathsf{e}) = \text{Benny}$
d. $I_0(\mathsf{f}) = \text{Frida}$

We will refer to the model $\langle D_0, I_0 \rangle$ as M_0. The interpretation function I_0 is responsible for mapping all of the non-logical constants to appropriate denotations based on D_0. It will map individual constants to elements of D_0, and one-place predicates to subsets of D_0. Two-place predicates are mapped to subsets of $D_0 \times D_0$, the Cartesian product of D_0 with itself. That is to say, each two-place predicate is mapped to a set of ordered pairs of individuals, where each individual is taken from D_0. However, D_0 itself does not contain any pairs. Predicates of higher arities are treated analogously. The same remarks apply to other models than D_0.

Since the interpretation function is a function, individual constants in that language are not ambiguous; they pick out exactly one object. This sets them apart from names in English such as *Björn*, which can refer to any individual with that name. However, not every individual object in a given model needs to have a corresponding individual constant in a given language. We could define a different model that includes any number of objects in the model's domain (say, Benny's nose) that are not named by any individual constant in the language. To illustrate, within the model $M_0 = \langle D_0, I_0 \rangle$ just defined, Benny's nose is not in D_0. Now consider a model $M_1 = \langle D_1, I_0 \rangle$ where I_0 is as before but D_1 is defined as $\{$ Agnetha, Björn, Benny, Frida, Benny's nose $\}$. Both models have the same interpretation function I_0. Now take a language L_0 which contains the non-logical constants $\mathsf{a}, \mathsf{b}, \mathsf{e}$, and f. Given this setup, there is no individual constant in L_0 that is mapped by I_0 to Benny's nose in either M_0 or M_1.

There is an important distinction between the objects themselves, which are not part of the formal language, and the non-logical constants that name these objects, which are. While Benny is a member of D_0 and D_1, and his nose is a member of D_1, neither Benny himself nor his nose is part of L_0.

Just as in propositional logic, we define a DENOTATION FUNCTION that coincides with I on basic expressions like individual constants and extends it to expressions of arbitrary complexity. This function will now depend not only on I but rather on M as a whole, and is therefore written $[\![\cdot]\!]^M$ rather than $[\![\cdot]\!]^I$:

(2) a. $[\![\mathsf{a}]\!]^{M_0}$ = Agnetha
b. $[\![\mathsf{b}]\!]^{M_0}$ = Björn
c. $[\![\mathsf{e}]\!]^{M_0}$ = Benny
d. $[\![\mathsf{f}]\!]^{M_0}$ = Frida

Here, we refer to the actual members of ABBA in our meta-language using their first names, and we write them capitalized and in ordinary type face. To echo Dowty et al. (1981): If it had been possible to persuade Agnetha Fältskog to come and occupy the right-hand side of the equation above for a moment, that would certainly have been preferable, but this is the closest we can come given that we are communicating with the reader via the printed page.

Starting in Chapter 6, we will define systems that relate English expressions to logical expressions, and thereby indirectly associate English expressions with denotations in the model. The mapping between expressions of the natural language (English) and their denotations (expressed in our meta-language) will thus be mediated by our logical representation language. So our ultimate theory will consist of two steps:

- *Björn* $\rightsquigarrow$ b (English to logic)
- $[\![\mathsf{b}]\!]^{M_0}$ = Björn (logic to denotation)

We follow the convention of italicizing natural language (e.g. English) expressions here and throughout. We say that *Björn* TRANS-

LATES TO b and that b (and, indirectly, *Björn*) DENOTES Björn. And similarly for other expressions. Again, as discussed in Chapter 1, the style of doing semantics we are adopting here is called INDIRECT INTERPRETATION. It differs from DIRECT INTERPRETATION in that we map English to some logic (the representation language) before assigning a denotation to natural language expressions.

Individual constants fall into the category of NON-LOGICAL CONSTANTS. This category includes not only individual constants but also some additional types of symbols that will be introduced below: predicate and function symbols. Why are they called 'non-logical' constants? In general, CONSTANTS are expressions whose denotation does not vary once a model has been specified. LOGICAL CONSTANTS are things like $\wedge$, whose denotation does not even vary from model to model; thus, $\wedge$ behaves according to its truth table across all models. The denotations of NON-LOGICAL CONSTANTS, on the other hand, depend on the model, and can vary from model to model. Later, we will introduce variables, whose denotation can vary even once a model has been specified.

The following rule ensures that the $[\![\cdot]\!]^M$ function tracks the I function on individual constants.

Semantic Rule: Non-logical constants
If α is a non-logical constant and $M = \langle D, I \rangle$, then:

$$[\![\alpha]\!]^M = I(\alpha)$$

Like ϕ and ψ in Section 3.2, α is a meta-variable. It is not part of the language L_0 we are defining but only part of the meta-language we are using to talk about that language. In the next section, we will see further applications of the semantic rule for non-logical constants.

4.1.2 Predication

4.1.2.1 Syntax of predication

True to its name, predicate logic has PREDICATES, along with individual constants. 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). Predicates that stand for properties are called UNARY PREDICATES. Predicates that stand for relations are called BINARY PREDICATES, TERNARY PREDICATES, or more generally, n-ARY PREDICATES depending on the number of entities they relate. A unary predicate combines with one term (sometimes called its ARGUMENT) to produce a statement that is true or false, depending on whether the individual denoted by the term has the property in question; if so, we also say that the predicate HOLDS of the individual. A binary predicate applies to two terms, a ternary predicate applies to three, and so on. In first-order logic, all predicates apply to individuals; in higher-order logic, predicates may also apply to other 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. As in the case of propositional letters, we will use interpretation functions to associate these symbols with denotations. The denotation of a unary predicate is a set of individuals (such as the set of female or Swedish or singing individuals). Predicate symbols combine with individual-denoting expressions to form ATOMIC FORMULAS. Like in propositional logic, formulas are true or false given a model. For example,

(3) swedish(a)

is a formula that consists of the predicate symbol Swedish and the constant a. Assume that we have a model that is defined in such a way that the predicate symbol Swedish denotes the set of individuals who are Swedish at the time of writing, and the constant a denotes Agnetha Fältskog, the ABBA singer. (We will make similar assumptions throughout.) Given this model, the formula in (3) denotes ⊤ if and only if Agnetha Fältskog is Swedish (which she is). Another way of talking about what is going on in (3) is that Swedishness is being 'predicated of' Agnetha Fältskog, so (3) can be called a PREDICATION. We will say more about the semantics of predications after we have laid out their syntax.

A BINARY PREDICATE denotes a relation between two individuals, and therefore combines with two terms. As an example of a binary predicate, we will use loves. A possible denotation for this predicate (in a given model) is the relation (the set of pairs) that contains a given pair of two individuals just in case the first loves the second in the actual world at the time of writing. A binary predicate combines with two terms:

(4) loves(a,b)

We say that the predicate loves HOLDS OF, APPLIES TO, or RELATES its two arguments. When translating transitive verbs like *love* from English to logic, the usual (and arbitrary) convention is to list the subject of an active sentence before its object. That is, we read (4) as "Agnetha loves Björn", not as "Björn loves Agnetha".

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. 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.

Predicates combine with the appropriate number of terms to form ATOMIC FORMULAS. As we have seen above, singer(a) is an atomic formula. Here a unary predicate singer 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 (4) 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.

(A summary of definitions like this can be found at the end of this section.)

The arity of a predicate is fixed in predicate logic. The arity of corresponding natural language expressions is much more free; for example, English allows the adjective *excited* to take a prepositional phrase complement but does not require it to.

(5) a. Agnetha is excited about Benny.
    b. Agnetha is excited.

In predicate logic, a predicate like Excited may only have a single arity; it cannot be both unary and binary. To represent the difference between the transitive and intransitive version of *excited* in English, one option would be to define two predicates, say, a unary predicate excited1 and a binary predicate excited2, which would produce well-formed formulas with the corresponding numbers of terms.

(6) a. excited1(a)

b. $\mathsf{excited2}(a, e)$

To capture how close in meaning these two predicates otherwise are, a theory could stipulate facts about how they relate to each other via constraints that are stipulated separately. (See MEANING POSTULATES below.)

Exercise 1. Give two examples of atomic formulas generated by the syntactic rule of Predication, choosing from the following individual constants and predicates:

- a and b are individual constants (a.k.a. 'names');
- singer and Swedish are one-place predicates;
- Knows and loves are two-place predicates.

4.1.2.2 Semantics of predication

So much for the syntax of predication. Now let us turn to the semantics. We will begin with some gossip. As it happens, in the 1970s, ABBA was composed of two married couples: Björn and Agnetha, as well as Frida and Benny. It therefore follows, by the principle that whenever two people are married to one another that they also love each other, that the sentences corresponding to the following formulas were true:

(7) a. loves(a,b)
b. loves(b,a)

(8) a. loves(e,f)
b. loves(f,e)

Now, as it happens, like all good things, both of the marriages eventually came to an end, and these four statements, concomitantly, ceased to be true (we assume). So far, we have only had one model, M_0. To distinguish between the way it was in the past, and how things later turned out, we will now edit it in two different ways: M_{THEN} corresponds to how it was back in the day, and M_{NOW} to how it is now. These two models share the same domain, D_0, but their interpretation functions will differ. We will define $M_{\text{THEN}} = \langle D_0, I_{\text{THEN}} \rangle$ and $M_{\text{NOW}} = \langle D_0, I_{\text{NOW}} \rangle$. (The subscripts $_{\text{THEN}}$ and $_{\text{NOW}}$ on these models and their components are meaningless; we just use them to label the two models in an easy-to-remember way.)

Relative to these two different models, the binary predicate loves has two different semantic values. Accordingly, we make the following assumptions about I_{THEN} and I_{NOW}:

(9) $I_{\text{THEN}}(\mathsf{loves}) =$
$\{\langle \text{Agnetha}, \text{Björn} \rangle, \langle \text{Björn}, \text{Agnetha} \rangle,$
$\langle \text{Frida}, \text{Benny} \rangle, \langle \text{Benny}, \text{Frida} \rangle\}$

(10) $I_{\text{NOW}}(\mathsf{loves}) = \{\}$

That is, back in the day, Agnetha and Björn loved each other, and so did Benny and Frida, but now, nobody loves each other. What we have done here is interpret the denotation of the binary predicate loves as a binary relation, that is, a set of ordered pairs.

Just as we did for individual constants, we need to make sure that $[\![\cdot]\!]^M$ function tracks the I function on predicates. We already have a rule that ensures this for all non-logical constants, so all we need to assume is that predicates count as non-logical constants. The semantic rule for non-logical constants given above then ensures that for any predicate α, $[\![\alpha]\!]^M = I(\alpha)$.

Just as we did for propositional letters, we will assume that the denotation of a formula like singer(a) is a truth value:

$$[\![\mathsf{singer(a)}]\!]^M = \mathsf{T} \text{ if } [\![\mathsf{a}]\!]^M \in [\![\mathsf{singer}]\!]^M, \text{ and } \mathsf{F} \text{ otherwise.}$$

The denotations of non-logical constants (including names and predicates) can differ across models. In some models, f denotes Frida, and in other models, it doesn't. In some models, Frida is in the denotation of $\textsf{singer}$, and in other models, she's not. Assuming the individual constant $\textsf{f}$ does denote Frida, the truth value of $\textsf{singer(f)}$ depends on whether Frida is in the denotation of $\textsf{singer}$. In general, for any given unary predicate π and any given term α, we would like the semantics of our language to ensure the following:

$$[\![\pi(\alpha)]\!]^M = \mathsf{T} \text{ if } [\![\alpha]\!]^M \in [\![\pi]\!]^M, \text{ and } \mathsf{F} \text{ otherwise.}$$

This can be read, "the semantic value of *pi* applied to *alpha* (with respect to model M) is T, if the semantic value of *alpha* (with respect to M) is an element of the semantic value of *pi* (with respect to M), and F otherwise." To put it somewhat more elegantly: "Relative to any given model, the predication of π upon α is true in that model if and only if the denotation of α in that model is a member of the set denoted by π in that model."

Our semantics should also ensure that the formula $\textsf{loves(a, b)}$ is true relative to M_{THEN} and false relative to M_{NOW}.

(11) a. $[\![\textsf{loves(a, b)}]\!]^{M_{\text{THEN}}} = \mathsf{T}$
because $\langle\text{Agnetha, Björn}\rangle \in [\![\textsf{loves}]\!]^{M_{\text{THEN}}}$
b. $[\![\textsf{loves(a, b)}]\!]^{M_{\text{NOW}}} = \mathsf{F}$
because $\langle\text{Agnetha, Björn}\rangle \notin [\![\textsf{loves}]\!]^{M_{\text{NOW}}}$

In general, for any binary predicate π, and any given terms α and β, our semantics should ensure:

$$[\![\pi(\alpha, \beta)]\!]^M = \mathsf{T} \text{ if } \langle [\![\alpha]\!]^M, [\![\beta]\!]^M \rangle \in [\![\pi]\!]^M, \text{ and } \mathsf{F} \text{ otherwise.}$$

This strategy can be generalized to predicates of arbitrary arity as follows:

Semantic Rule: Predication
If π is a predicate of arity n and $\alpha_1, ..., \alpha_n$ is a sequence of terms, then:

$$[\![\pi(\alpha_1, ... \alpha_n)]\!]^M = \mathsf{T} \text{ if } \langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M \rangle \in [\![\pi]\!]^M, \text{ and } \mathsf{F} \text{ otherwise.}$$

To make sure this works as expected in the unary case, we adopt the convention that $\langle [\![\alpha_n]\!]^M \rangle = [\![\alpha_n]\!]^M$.

Exercise 2. Suppose we have a particular model $M_2 = \langle D_2, I_2 \rangle$. Let $D_2 = \{\text{Agnetha, Björn, Benny, Frida}\}$. Suppose that in M_2, everybody loves themselves and nobody loves anybody else, and the binary predicate loves denotes the 'love' relation. What is then the value of $I_2(\text{loves})$? Specify the relation as a set of ordered pairs.

Exercise 3. Assume a model

$$M_3 = \langle D_3, I_3 \rangle$$

where D_3 contains Abelard and Eloise:

$$D_3 = \{\text{Abelard, Eloise}\}$$

and I_3 is defined as follows:

$$I_3 = \begin{bmatrix} \text{a} & \to & \text{Abelard} \\ \text{e} & \to & \text{Eloise} \\ \text{female} & \to & \{\text{Eloise}\} \\ \text{scholar} & \to & \{\text{Abelard, Eloise}\} \\ \text{loves} & \to & \{\langle\text{Abelard,Eloise}\rangle, \langle\text{Eloise,Abelard}\rangle, \langle\text{Eloise,Eloise}\rangle\} \\ \text{teacher} & \to & \{\langle\text{Abelard,Eloise}\rangle\} \end{bmatrix}$$

For each of the following formulas, use the semantic rule of Predication to determine its semantic value in model M_3:

(a) teacher(a,e)

(b) teacher(e,a)

(c) loves(a,a)

(d) scholar(a)

(e) female(a)

On a philosophical note: As in the case of propositional letters, we can think of predicate symbols either as carrying intrinsic meanings, or as being devoid of any intrinsic meaning apart from what the model supplies. On the first view, the model (and more specifically, its interpretation function) corresponds to a possible world (or way things could be), and its specification of denotations for each of the predicates constitutes a specification of how that world is. On this view, the predicate Sings has some intrinsic meaning. On the second view, the model supplies an otherwise meaningless symbol with a denotation, and the predicate Sings has no intrinsic meaning. The model associates it with a set of individuals, and there is nothing more to it than that.

Because a model assigns sets to different predicate symbols independently of one another, the two perspectives are equivalent only as long as one assumes that the intrinsic meanings of different predicate symbols are independent of one another. By contrast, if we were to take the predicate symbols bachelor and married in a given language to stand for the properties of being a bachelor and of being married, a model that maps the two predicate symbols to overlapping sets would not correspond to any possible circumstance (since it's impossible for a bachelor to be married). MEANING POSTULATES are a mechanism for limiting attention to just those models in which these formulas are true; these are called ADMISSIBLE MODELS. A typical meaning postulate is a formula that might correspond to a sentence like "No bachelor is married" or "It is not both Tuesday and Wednesday". Even if one takes the view that predicate symbols are inherently meaningless, the use of meaning postulates can be seen as imbuing these symbols with some degree of meaning, at least enough so that they interact with other symbols in the way that would be expected if they corresponded to particular concepts that the theorist has in mind.

4.1.3 Functions

Recall that a TERM is an expression that denotes an individual in the domain. So far, the only kind of term that we have seen are individual constants. But it is also possible to form syntactically complex terms using FUNCTION SYMBOLS. A function symbol denotes a function, in the sense defined in Chapter 2. In first-order logic, a function symbol denotes a function from individuals (or n-tuples of individuals) to individuals. This means that the function has to associate every individual in the relevant domain with a value (which is also an individual in that domain), and provide a unique output for each input. In this, function symbols contrast with predicate symbols, which denote sets.

Confining our attention to models where every individual in the domain has exactly one spouse,

$$\mathsf{spouseOf(e)}$$

could be used to denote the spouse of Benny. (We continue to assume throughout that e denotes Benny.) It is not used to make a claim about Benny, like a predicate does, and the entire expression does not express something that can be true or false, as in the case of a predicate. Rather, this expression denotes a particular individual.

Syntactically, function symbols combine with terms. Combinations of function symbols with terms are called COMPLEX TERMS. Since complex terms are terms, they can appear in all the same positions as individual constants and other terms. For instance, complex terms can fill the first argument slot of a binary relation:

$$\mathsf{loves(spouseOf(b), b)}$$

This formula can be read as saying that Benny's spouse loves him (Benny).

Complex terms can even combine with function symbols again. For example, the following is a complex term that can be used to refer to Benny's spouse's spouse, or in other words, Benny himself:

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

Predicates and functions are easy to confuse with each other, because they both combine with terms in parentheses. To distinguish between them, this textbook uses the following conventions: functions that output individuals end in of unless we explicitly specify otherwise. (These conventions are just what we do in this book. There is no standard set of conventions of that sort across the field.) This way, all terms (both individual constants and complex terms formed with functions) start with lowercase letters. As with predicate and name symbols, we follow the convention that any sequence of numbers or letters or underscores may follow the initial letter, but no spaces.

Functions, like predicates, have a particular arity. The function spouseOf has arity 1 (i.e, it is a UNARY FUNCTION). As an example of a function with arity 2, suppose that in model M_1, the expression tallerOneOf denotes the function that takes two arguments and returns whichever one is taller at the time of writing. Assuming Frida is currently taller than Agnetha,

$$\mathsf{tallerOneOf}(\mathsf{a},\mathsf{f})$$

denotes Frida in M_1. In general:

Syntactic rule: Complex terms
Given any function γ with arity n, then:

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

is a term, where $\alpha_1,...,\alpha_n$ is a sequence of expressions that are themselves terms.

The denotation of function symbols is specified by the interpretation function I. Just like individual constants and predicate symbols, function symbols are considered non-logical constants; therefore, their denotation is derived from I according to the same rule as individual constants and predicate symbols. However, function symbols combine with terms in a slightly different manner from the way predicate symbols do. The denotation of a function symbol applied to a term is the result of applying the function denoted by the function symbol to the denotation of the term, as opposed to checking for set membership. For example, suppose that in model M_1, $\mathsf{spouseOf}$ denotes a function that returns Frida when given the individual Benny as an argument. Then $\mathsf{spouseOf(e)}$ denotes Frida, i.e., $[\![\mathsf{spouseOf(e)}]\!]^{M_1}$ = Frida. In general:

Semantic Rule: Complex terms
If γ is a unary function symbol, and α is a term, then:

$$[\![\gamma(\alpha)]\!]^M = [\![\gamma]\!]^M([\![\alpha]\!]^M)$$

The preceding formula can be read, "the semantic value of gamma applied to alpha (in model M) is equal to the semantic value of gamma (in M) applied to the semantic value of alpha (in M)." Observe that we are using parentheses around alpha both in the object language (in this context, the formal representation language) and the meta-language here. The parentheses in the object language (on the left) help to create a complex term consisting of the function symbol and its argument. The parentheses in the meta-language (on the right) signify the application of the denoted function to the actual individual denoted by the term.

A binary function symbol like $\mathsf{tallerOneOf}$ combines with two terms. In general:

If γ is a binary function, and α and β are terms, then:

$$[\![\gamma(\alpha,\beta)]\!]^M = [\![\gamma]\!]^M(\langle[\![\alpha]\!]^M, [\![\beta]\!]^M\rangle)$$

This definition can be generalized to accommodate functions of arbitrary arity:

Semantic Rule: Complex terms
If γ is a function of arity n, and $\alpha_1, ..., \alpha_n$ is a sequence of n terms, then:

$$[\![\gamma(\alpha_1, ..., \alpha_n)]\!]^M = [\![\gamma]\!]^M(\langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M \rangle)$$

Exercise 4. Which of the following are well-formed formulas? Assume happy is a unary predicate, $\mathsf{spouseOf}$ is a unary function, and $\mathsf{tallerOneOf}$ is a binary function (you can assume that it returns the taller one of two individuals).

(a) $\mathsf{happy}(\mathsf{spouseOf}(\mathsf{a}))$

(b) $\mathsf{happy}(\mathsf{spouseOf}(\mathsf{a}, \mathsf{e}))$

(c) $\mathsf{happy}(\mathsf{tallerOneOf}(\mathsf{a}, \mathsf{e}))$

(d) $\mathsf{happy}(\mathsf{spouseOf}(\mathsf{a}), \mathsf{spouseOf}(\mathsf{e}))$

4.1.4 Identity

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. 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.

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. The following are all atomic formulas:

$$\mathsf{a} = \mathsf{e}$$

$$\mathsf{spouseOf}(\mathsf{a}) = \mathsf{b}$$

$$\mathsf{f} = \mathsf{tallerOneOf}(\mathsf{f}, \mathsf{a})$$

The following rule dictates that any two terms, simple or complex, can be joined in this way to form an atomic formula:

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

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.

Semantically, the interpretation of identity is independent of the model. Unlike predicates, but similarly to connectives, the interpretation of the symbol does not vary from model to 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.

Exercise 5. For each of the following, say whether it is well-formed, and give a paraphrase in English. (For the ones that are not well-formed, the paraphrase in English might sound like nonsense, and that's OK.)

(a) $\mathsf{spouseOf}(\mathsf{a}) = \mathsf{spouseOf}(\mathsf{e})$

(b) $\mathsf{spouseOf}(\mathsf{a}) \leftrightarrow \mathsf{spouseOf}(\mathsf{e})$

(c) $\mathsf{happy}(\mathsf{spouseOf}(\mathsf{a})) \leftrightarrow \mathsf{happy}(\mathsf{spouseOf}(\mathsf{e}))$

(d) $\mathsf{happy}(\mathsf{spouseOf}(\mathsf{a})) = \mathsf{happy}(\mathsf{spouseOf}(\mathsf{e}))$

It is common to extend predicate logic with other binary predicates taken from mathematics as well, such as q, $\leq$, 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 $\mathsf{a} = \mathsf{b}$ instead of $= (\mathsf{a}, \mathsf{b})$. 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 10, we will add a predicate standing for the parthood relation.

Section summary

To summarize, our formal language so far is made up of basic and complex expressions. We have the following types of basic expressions, which are all non-logical constant symbols in our language.

CATEGORY	EXAMPLE
individual constants	a
unary predicates	singer
binary predicates	loves
function symbols	$\mathsf{spouseOf}$

In addition to our basic expressions, we have syntactic and semantic rules for creating complex terms using function symbols, and two ways of creating atomic formulas: predication and identity. Both of these types have a corresponding semantic rule. On top of this we retain all of the syntactic and semantic rules from propositional logic, including negation and the rules for creating complex formulas using binary connectives.

Exercise 6. Let us consider a model $M_4 = \langle D_4, I_4 \rangle$ with domain D_4 consisting only of two individuals: Abelard and Eloise. Let us assume that among our basic expressions we have names for both Abelard and Eloise (say a and e respectively), as well as the unary predicates scholar, male, and female, binary predicates loves and younger, and the function terms spouseOf and selfOf. The intended interpretation of selfOf is a function that applies to an individual and returns that very individual that was given as input as output. Fill in the missing values in the interpretation function, according to what you think they should be based on the constant symbol:

$$I_4 = \begin{bmatrix} \mathsf{a} & \rightarrow & \text{Abelard} \\ \mathsf{e} & \rightarrow & \text{Eloise} \\ \mathsf{female} & \rightarrow & \{\text{Eloise}\} \\ \mathsf{male} & \rightarrow & \\ \mathsf{scholar} & \rightarrow & \{\text{Abelard, Eloise}\} \\ \mathsf{loves} & \rightarrow & \{\langle\text{Abelard,Eloise}\rangle, \langle\text{Eloise,Abelard}\rangle, \langle\text{Eloise,Eloise}\rangle\} \\ \mathsf{younger} & \rightarrow & \{\langle\text{Eloise,Abelard}\rangle\} \\ \mathsf{spouseOf} & \rightarrow & \{\langle\text{Abelard,Eloise}\rangle, \langle\text{Eloise,Abelard}\rangle\} \\ \mathsf{selfOf} & \rightarrow & \end{bmatrix}$$

Exercise 7. Fill in the following table based on the rules of L_0.

	Term or formula?	$[\![\cdot]\!]^{M_4}$	**Semantic Rule(s)**

spouseOf(a)	term	Eloise	Complex terms
female(a)	formula	F	Atomic formulas
male(a, e)	not well-formed!	N/A	N/A
younger(a, e)			
younger(spouseOf(e), e)			
loves(a, selfOf(a))			
spouseOf(spouseOf(e))			
scholar(selfOf(selfOf(a)))			
scholar(a, selfOf(a))			
younger(selfOf(selfOf(a)))			

In the first labelled column, state whether the expression is a term, a formula, or not well-formed. In the second, give the semantic value relative to the model you designed in exercise 6. In the third, indicate the semantic rule(s) you used to derive the semantic value in the second column.

Exercise 8. This exercise has seven parts, labelled (a)-(g) below.

One of the following arguments is valid and the other is not. The valid one, of course, is (12).

(12) a. Ben and Jerry are brothers.
b. ∴ Ben is Jerry's brother.

(13) a. Ben and Jerry are computers.
b. $\not\therefore$ Ben is Jerry's computer.

Relational nouns are nouns that denote two-place predicates (a.k.a. 'binary *relations*'); *sortal nouns* are ones that denote one-place predicates. When N is a relational noun, representable as a binary relation R, a sentence of the form 'x and y are Ns' can be translated into predicate logic as:

$$[R(x,y) \wedge R(y,x)]$$

In other words, the construction asserts that x and y stand in the relation to *each other*. (This explains why *Jerry and Ben are brothers* is unremarkable but *??Jerry and Sheila are brothers* is quite jarring, under conventional assumptions about what names signal about the gender of the referent.) In this sense, the construction expresses a *reciprocal* relation between x and y.

When N is a sortal noun, representable by predicate P, a sentence of the form 'x and y are Ns' can be translated into predicate logic as:

$$[P(x) \wedge P(y)]$$

In other words, the construction says that both x and y have the property in question. Let's think about how this theory can explain the contrast in the preceding question. First, we need to decide how to classify the nouns *brother* and *computer*.

(a) Should we classify *brother* as sortal or relational?
(b) How about *computer*?

With these assumptions, let us now provide translations into predicate logic, starting with the (a) sentences. Use $\mathsf{Brother}$ as your translation for *brother* and $\mathsf{Computer}$ for *computer*, and use the individual constants b and j as your translations for *Ben* and *Jerry* respectively. Make sure that your formulas are well-formed according to our syntax rules!

(c) *Ben and Jerry are brothers.*

(d) *Ben and Jerry are computers.*

Possessive statements of the form 'x is y's N' must be analyzed slightly differently depending on whether N is sortal or relational. If N is relational, and denotes the binary relation R, then 'x is y's N' just expresses that x and y stand in the relation R:

$$R(X, Y)$$

On the other hand, if N is sortal, then 'x is y's N' expresses (i) that x is an N and (ii) that some kind of possessive relation holds between x and y. Let's use the two-place predicate poss to denote this possession relation (which must be general enough to cover a broad range of more specific possessive relations that may be implied in context). Let's use $\mathsf{poss}(y, x)$ to signify 'y possesses x'. So for a sortal noun N translated as one-place predicate P, 'x is y's N' would be translated as:

$$[P(x) \wedge \mathsf{poss}(y, x)]$$

With this in mind, give a translation for the following sentences:

(e) *Ben is Jerry's brother.*
(f) *Ben is Jerry's computer.*

Now we are in a position to derive the fact that (12) and (13) above differ in validity. In general, arguments of the following form are valid:

$$[\phi \wedge \psi]$$
$$\therefore \phi$$

(g) Based on this fact (called *conjunction elimination*) and the translations into logic that we have given, explain why the inference is valid in one example but not the other. (Make sure to address both examples.)

4.2 Quantification

Consider this argument:

(14) Aristotle taught Alexander the Great.
Alexander the Great was a king.
$\therefore$ 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. A VARIABLE is a symbol that is just like a constant symbol except that the model does not specify the individual it stands for (unlike in the case of constants). We will use the symbols x, y, and z, and similar ones that we will introduce later, as variables, and never as constants. The condition that the object should satisfy may be written as follows:

(15) $[\mathsf{taught}(\mathsf{aristotle}, x) \wedge \mathsf{king}(x)]$

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). To make the claim that there is some individual that satisfies this condition, we may use the EXISTENTIAL QUANTIFIER, $\exists$.

(16) $\exists x[\mathsf{taught}(\mathsf{aristotle}, x) \wedge \mathsf{king}(x)]$

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 $\mathsf{aristotle}$ 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 (16), we would have expressed the claim that *everything* satisfies the condition in (15). 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:

(17) $\forall x[\mathsf{philosopher}(x) \rightarrow \mathsf{studies}(x, \mathsf{aristotle})]$

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:

(18) $\forall x[\mathsf{philosopher}(x) \wedge \mathsf{studies}(x, \mathsf{aristotle})]$

This says, "For all x, x is a philosopher and x studies Aristotle" – in other words, "Everything/everyone is an 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:

(19) $[\textsf{philosopher}(x) \rightarrow \textsf{studies}(x, \textsf{aristotle})]$

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

$$\textsf{philosopher}(x)$$

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 $[\textsf{philosopher}(x) \rightarrow \textsf{studies}(x, \textsf{aristotle})]$. 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. And this is exactly what *Every philosopher studies Aristotle* says.

Now consider the following formula:

(20) $\forall x[\textsf{linguist}(x) \rightarrow \exists y[\textsf{philosopher}(y) \wedge \textsf{admires}(x, y)]]$

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:

(21) $\exists y[\mathsf{philosopher}(y) \wedge \forall x[\mathsf{linguist}(x) \rightarrow \mathsf{admires}(x, y)]]$

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:

(22) 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.

Exercise 9. For each of the following formulas, say (i) how you would read the formula aloud, using phrases like 'for all x' and 'there exists an x such that' and (ii) give a natural paraphrase in English.

(a) $\forall x.\mathsf{friendly}(x)$

(b) $\forall x[\mathsf{friendly}(x) \wedge \mathsf{happy}(x)]$

(c) $\exists x[\mathsf{friendly}(x) \wedge \mathsf{happy}(x)]$

(d) $\exists x[\mathsf{friendly}(x) \vee \mathsf{happy}(x)]$

(e) $\forall x[\mathsf{friendly}(x) \rightarrow \mathsf{happy}(x)]$

(f) $\forall x.\neg\mathsf{friendly}(x)$

(g) $\exists x.\neg\mathsf{friendly}(x)$

(h) $\neg\exists x.\mathsf{friendly}(x)$

(i) $\forall x.\exists y.\mathsf{loves}(y, x)$

Exercise 10. For each of the following sentences, say which of the formulas above it matches (if any). (In some cases, the sentence might match two formulas.)

(a) Somebody is friendly and happy.

(b) Everybody is friendly and happy.

(c) Everybody who is friendly is happy.

(d) Nobody is friendly.

(e) Somebody is not friendly.

(f) Somebody is friendly or happy.

(g) Everybody loves somebody.

(h) Somebody loves everybody.

Exercise 11. 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) $\forall x[\mathsf{cellist}(x) \wedge \mathsf{smokes}(x)]$

Exercise 12. Express the following sentences in a variant of L_{Pred} that you have augmented with any necessary basic expressions:

(a) There is a red car.

(b) All cars are red or green.

(c) No car is blue.

(d) Alan dislikes all cars.

Feel free to add as many non-logical constants as you need.

Exercise 13. Express the following sentences. In some cases, there may be quantifier scope ambiguity; in that case, give a representation corresponding to both interpretations.

(a) Every even number is divisible by two.

(b) Everything has a reason.

(c) Something is the reason for everything.

(d) Every human being has at least two mothers.

(e) All fathers are older than their children.

(f) If a man is a philosopher then he is mortal.

(g) Some statues are not of marble.

(h) All statues are not of marble.

(i) He who sins sleeps badly.

Feel free to add as many non-logical constants as you need.

Note: This exercise is extremely challenging!

Now let us start to give a formal definition of the syntax of a language with variables and quantifiers. We will refer to this language as $\mathrm{L}_{\mathrm{Pred}}$. 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. We use x as an abbreviation for x_0, and similarly for y and z. (It is not good practice to mix the abbreviated and non-abbreviated versions of the same variable, so x and x_0 should never both be used within the same formula.) We will also add new formation rules for the universal quantifier $\forall$ and the existential quantifier $\exists$.

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 meta-variables; that is to say, they belong to our meta-language (English with some mathematical bits mixed in). They stand for variables (such as x) and formulas (such as $\mathsf{happy}(x)$) that can occur in formulas of the logic, but they themselves cannot occur in any formulas. For example, $[\forall x.\mathsf{happy}(x)]$ is a well-formed formula according to these rules (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. $\forall x[\mathsf{happy}(x) \to \mathsf{friendly}(x)]$. 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.

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,

(23) $\forall x.\mathsf{happy}(x)$

is true in a model M if (and only if) no matter which individual we assign as the interpretation of x,

(24) $\mathsf{happy}(x)$

is true. Likewise, informally,

(25) $\exists x.\mathsf{happy}(x)$

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:

(26) a. Napoleon had all the properties of a good general.
b. No two distinct objects have the same properties.

Example (26b) is often referred to as the law of the Identity of Indiscernibles, 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:

$$\forall x[\mathsf{happy}(x) \to \exists y . \mathsf{Likes}(x, y)]$$

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:

$$[\mathsf{happy}(x) \to \exists y . \mathsf{Likes}(x, y)]$$

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.

Variables should not be confused with constants; they are the opposite of constants. While constants get their interpretation from the interpretation function (which is part of the model), variables get their interpretation from the assignment function (which is not part of the model, and which is acted on by quantifiers). The model continues to consist of just a domain of individuals and an interpretation function.

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:

$$g_1 = \begin{bmatrix} x & \rightarrow & \text{Agnetha} \\ y & \rightarrow & \text{Benny} \\ z & \rightarrow & \text{Benny} \\ \ldots & & \end{bmatrix} \qquad g_2 = \begin{bmatrix} x & \rightarrow & \text{Benny} \\ y & \rightarrow & \text{Björn} \\ z & \rightarrow & \text{Benny} \\ \ldots & & \end{bmatrix}$$

The domain of an assignment function is the set of variables.

In order to interpret an expression like $\text{happy}(x)$, we need *both a model and an assignment function*: The model tells us who is happy, and the assignment function determines a value for 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 L_{Pred}: Variables
$[\![u]\!]^{M,g} = g(u)$

For example, $[\![x]\!]^{M,g_1} = g_1(x) = \text{Agnetha}$, and $[\![x]\!]^{M,g_2} = g_2(x) = \text{Benny}$ (regardless of our choice of model M).

Exercise 14. 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. $[\![\mathsf{happy}]\!]^{M,g_1}$ will be the same as $[\![\mathsf{happy}]\!]^{M,g_2}$ 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

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

depends on the value that is assigned to x. Whether $\mathsf{happy}(x)$ is true or not depends on how x is interpreted, and this is given by the assignment function.

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. 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:

$$g_1 = \begin{bmatrix} x & \rightarrow & \text{Agnetha} \\ y & \rightarrow & \text{Benny} \\ z & \rightarrow & \text{Benny} \\ \ldots & & \end{bmatrix}$$

$g_1[y \mapsto \text{Björn}]$ would be as follows:

$$g_1[y \mapsto \text{Björn}] = \begin{bmatrix} x & \rightarrow & \text{Agnetha} \\ y & \rightarrow & \text{Björn} \\ z & \rightarrow & \text{Benny} \\ \ldots & & \end{bmatrix}$$

We changed it so that y maps to Björn and kept everything else the same.

Exercise 15.

(a) What is $g_1[z \mapsto \text{Björn}](x)$? (I.e., what does $g_1[z \mapsto \text{Björn}]$ assign to x?)

(b) What is $g_1[z \mapsto \text{Björn}](y)$?

(c) What is $g_1[z \mapsto \text{Björn}](z)$?

With this terminology, we can give the following official semantics for $\exists x\,.\,\mathsf{happy}(x)$:

$$[\![\exists x\,.\,\mathsf{happy}(x)]\!]^{M,g} = \mathsf{T} \text{ iff there is an individual } k \in D \text{ such that: } [\![\mathsf{happy}(x)]\!]^{M,g[x\mapsto k]} = \mathsf{T}.$$

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:

$$[\![\phi]\!]^{M,g[x\mapsto k]} = \mathsf{T}$$

Now, if we wanted 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 $k \in D$:

$$[\![\phi]\!]^{M,g[v \mapsto k]} = \mathsf{T}$$

4.2.1 Syntax of L_{Pred}

Let us now summarize the syntactic rules of our language. (We will not list every single name, function, and predicate, but rather only list a few examples.)

1. **Basic Expressions**
   - **Individual constants:** a, b, e, f, ...
   - **Individual variables:** x_n, y_n, and z_n for every natural number n;
     (x is an optional abbreviation for x_0 which must be used consistently throughout a formula if it is used at all; similarly for y and z)
   - **Function symbols**
     - Unary: $\mathsf{spouseOf}$, ...
     - Binary: $\mathsf{tallerOneOf}$, ...
   - **Predicate symbols**
     - Unary: happy, ...
     - Binary: loves, ...
2. **Terms**
   - Every individual constant is a term.

- Every individual variable is a term.
- If π is a function symbol of arity n, and $\alpha_1, \ldots, \alpha_n$ are terms, then $\pi(\alpha_1, \ldots, \alpha_n)$ is a term.[3]

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.[4]
   - **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:

---

[3] Special cases:

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

[4] 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 formula.

- $[\forall u . \phi]$ ‘for all u: ϕ’
- $[\exists u . \phi]$ ‘there exists a u such that ϕ’

Variables are either FREE or BOUND in a given formula. Whether a variable is free or bound is defined syntactically as follows:

- In an atomic formula, any variable is free.
- The free variables in ϕ are also free in $\neg\phi$, and the free variables in ϕ and ψ are free in $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \rightarrow \psi]$, and $[\phi \leftrightarrow \psi]$.
- All of the free variables in ϕ are free in $[\forall u . \phi]$ and $[\exists u . \phi]$, except for u, and every occurrence of u in ϕ is bound in the quantified 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.[5]

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:

$$[\forall x[\mathsf{linguist}(x) \rightarrow [\exists y . \mathsf{admires}(x, y)]]]$$

[5]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, $\mathsf{Swedish}(\mathsf{a})$ is equivalent to $[\forall x . \mathsf{Swedish}(\mathsf{a})]$ and to $[\exists x . \mathsf{Swedish}(\mathsf{a})]$. These formulas are all true (on the intended interpretation) just in case Agnetha is Swedish. To put it differently, if it is true that Agnetha is Swedish, then it is also true of every individual, and of some individual, that Agnetha is Swedish.

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.

4.2.2 Semantics of $\mathrm{L}_{\mathrm{Pred}}$

Now for the semantics of $\mathrm{L}_{\mathrm{Pred}}$. 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. **Basic Expressions**
   - 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. **Complex terms**
   - If π is a function of arity n, and $\alpha_1, ..., \alpha_n$ is a sequence of n terms, then:

$$[\![\pi(\alpha_1, ..., \alpha_n)]\!]^{M,g} = [\![\pi]\!]^{M,g}(\langle [\![\alpha_1]\!]^{M,g}, ..., [\![\alpha_n]\!]^{M,g} \rangle)$$

3. **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.[6]
   - **Identity**
     If α and β are terms, then

$$[\![\alpha = \beta]\!]^{M,g} = \mathsf{T} \text{ if } [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g},$$

[6]Special cases:

– When π is a predicate of arity 1, then:

$$[\![\pi(\alpha)]\!] = \mathsf{T} \text{ if } [\![\alpha]\!]^{M,g} \in [\![\pi]\!]^{M,g} \text{ and F otherwise.}$$

– When π is a function of arity 2, then:

$$[\![\pi(\alpha, \beta)]\!] = \mathsf{T} \text{ if } \langle [\![\alpha]\!]^{M,g}, [\![\beta]\!]^{M,g} \rangle \in [\![\pi]\!]^{M,g} \text{ and F otherwise.}$$

and F otherwise.

4. **Negation**
   - $[\![\neg\phi]\!]^{M,g} = \mathsf{T}$ if $[\![\phi]\!]^{M,g} = \mathsf{F}$, and F otherwise.

5. **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.)

6. **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.

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

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

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 $\forall x.\mathsf{happy}(x)$ has the same value relative to every assignment function.

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.

Exercise 16. Consider the following formulas.

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

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

(c) $\mathsf{happy}(\mathsf{m}, \mathsf{m})$

(d) $\neg\neg\mathsf{happy}(\mathsf{n})$

(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 L_{Pred}?

(ii) Of the ones that are well formed in L_{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 17. Consider the following model $M_f = \langle D, I_f \rangle$, where everybody is happy:

$$I_f(\textsf{happy}) = \{\text{Benny}, \text{Björn}, \text{Agnetha}\}$$

Assume that $g_{\text{Benny}} = g_1[x \mapsto \text{Benny}]$ in the problems below.

(a) What is $[\![x]\!]^{M_f, g_{\text{Benny}}}$? Apply the L_{Pred} semantic interpretation rule for variables.

(b) What is $[\![\textsf{happy}]\!]^{M_f, g_{\text{Benny}}}$? Apply the relevant L_{Pred} semantic interpretation rule.

(c) Which semantic interpretation rule do you need to use in order to put the denotations of $\textsf{happy}$ and x together, and compute the denotation of $\textsf{happy}(x)$?

(d) Using the rule you identified in your answer to the previous question, explain carefully why $[\![\textsf{happy}(x)]\!]^{M_f, g_{\text{Benny}}} = \mathsf{T}$.

Exercise 18. Consider the following four assignment functions.

$$g_{ae} = \begin{bmatrix} x & \rightarrow & \text{Abelard} \\ y & \rightarrow & \text{Eloise} \end{bmatrix} \qquad g_{ea} = \begin{bmatrix} x & \rightarrow & \text{Eloise} \\ y & \rightarrow & \text{Abelard} \end{bmatrix}$$

$$g_{ee} = \begin{bmatrix} x & \rightarrow & \text{Eloise} \\ y & \rightarrow & \text{Eloise} \end{bmatrix} \qquad g_{aa} = \begin{bmatrix} x & \rightarrow & \text{Abelard} \\ y & \rightarrow & \text{Abelard} \end{bmatrix}$$

For each of the following expressions, give the semantic value of the expression relative to the model M defined in Exercise 6 and each of the four assignment functions, using the syntax and semantics of L_{Pred}. In other words, say for each expression α what $[\![\alpha]\!]^{M,g}$ is, for each given assignment function g.

Give your answer in the form of a table, with columns labelled g_{ae}, g_{ea}, g_{ee}, and g_{aa}.

(a) x

(b) y

(c) a

(d) $\mathsf{spouseOf}(x)$

(e) $\mathsf{female}(x)$

(f) $[\mathsf{female}(x) \wedge \mathsf{scholar}(x)]$

(g) $[\mathsf{female}(x) \rightarrow \mathsf{scholar}(x)]$

(h) $\mathsf{teacher}(x, y)$

(i) $\exists y.\mathsf{teacher}(x, y)$

(j) $\exists x \exists y.\mathsf{teacher}(x, y)$

(k) $\mathsf{teacher}(\mathsf{a}, y)$

(l) $\exists y.\mathsf{teacher}(\mathsf{a}, y)$

(m) $\mathsf{loves}(x, \mathsf{spouseOf}(x))$

(n) $\forall x . \mathsf{loves}(x, \mathsf{spouseOf}(x))$

(o) $\mathsf{loves}(x, y)$

(p) $\forall x . \mathsf{loves}(x, y)$

(q) $\exists y \forall x . \mathsf{loves}(x, y)$

(r) $\mathsf{teacher}(x, y) \to \mathsf{male}(x)$

(s) $\forall y [\mathsf{teacher}(x, y) \to \mathsf{male}(x)]$

(t) $\forall x \forall y [\mathsf{teacher}(x, y) \to \mathsf{male}(x)]$

Exercise 19. Let g be defined such that $x \mapsto$Frida, $y \mapsto$Benny, and $z \mapsto$Björn, and suppose that in M_2, everybody loves themselves and nobody loves anybody else, and the binary predicate loves denotes this love relation. Assume that f denotes Frida.

(a) Calculate:

(i) $[\![x]\!]^{M_2, g}$

(ii) $[\![\mathsf{f}]\!]^{M_2, g}$

(iii) $[\![\mathsf{loves}]\!]^{M_2, g}$

(iv) $[\![\mathsf{loves}(x, \mathsf{f})]\!]^{M_2, g}$

(b) List all of the value assignments that are exactly like g except possibly for the individual assigned to x, and label them $g_1 ... g_n$.

(c) For each of those value assignments g_i in the set $\{g_1, ..., g_n\}$, calculate $[\![\mathsf{loves}(x, \mathsf{f})]\!]^{M_2, g_i}$.

(d) On the basis of these and the semantic rule for universal quantification calculate $[\![\forall x\,.\,\mathsf{loves}(x,\mathsf{f})]\!]^{M_2,g}$ and explain your reasoning.

Exercise 20. If a formula has free variables then it may well be true with respect to some assignments and false with respect to others. Give an example of two variable assignments g_i and g_j such that $[\![\mathsf{loves}(x,\mathsf{f})]\!]^{M_2,g_i} \neq [\![\mathsf{loves}(x,\mathsf{f})]\!]^{M_2,g_j}$.

Exercise 21. In the Algonquian language Passamaquoddy (spoken in Maine, United States, and New Brunswick, Canada), voice marking on the verb can affect which scope readings are available for quantifiers (Bruening, 2001, 2008). For example, (27) and (28) differ in voice-marking and are true in different circumstances.

(27) Skitap psite 'sakolon-a puhtaya.
man all hold-DIRECT bottles
'A man is holding all the bottles.'

(28) Psite puhtayak 'sakolon-ukuwal peskuwol skitapiyil.
all bottles hold-INDIRECT one man
'All of the bottles are held by some man.'

(The morphological glosses have been simplified.)

In (27), the verb is in direct voice, and the agent of the verb *hold* corresponds to the bare noun *skitap* 'man', interpreted as an indefinite ('a man'). The patient (the thing being held) corresponds to *puhtaya* 'bottle', which is associated with the universal quantifier *psite* 'all'. Speakers of Passamaquoddy judge this sentence to be true in the situation on the right in Figure 4.1, but not in the situation on the left. (Images created by Benjamin

Figure 4.1: Left: A situation where each man is holding a different bottle. Right: A situation where one man is holding all of the bottles. (See exercise 21.)

Bruening for the Scope Fieldwork Project; see `http://udel.edu/~bruening/scopeproject/materials.html`.)

In (28), the verb is in indirect voice, and again the agent corresponds to an indefinite noun phrase meaning 'a man', and the patient corresponds to 'all bottles'. This version of the sentence can be interpreted in two ways, one where the picture on the left in Figure 4.1 makes it true, and one where the picture on the right makes it true.

(a) Write out representations in L_{Pred} for the two possible scope interpretations.

(b) Given that the version in direct voice is true only in the situation on the right, which of the two scope interpretations is correct for direct voice?

(c) More generally, what does this contrast suggest about how voice affects scope interpretation in Passamaquoddy?

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 modeled 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.

Consider the sentence *John loves Mary*, which might be translated into $\mathrm{L}_{\mathrm{Pred}}$ as:

$$\mathsf{loves}(\mathsf{j}, \mathsf{m})$$

Some parts of the sentence *John loves Mary* can be straightforwardly mapped into expressions in $\mathrm{L}_{\mathrm{Pred}}$, but others do not map onto self-contained chunks. For example, we might say that (relative to a given model) the English name *Mary* picks out a particular element of the domain, namely Mary. So it makes sense to translate the English name *Mary* as an individual constant, such as m, as this is the sort of denotation that individual constants have. The English verb *loves* could be thought of as denoting a binary relation (a set of ordered pairs of individuals in the domain), the sort of thing denoted by a binary predicate. Let us therefore assume that loves is a binary predicate and that the verb maps to it. But what does a verb phrase like *loves Mary* map onto? Your intuition as a theorist might tell you that it translates to a formula with an empty slot:

(1) $\mathsf{loves}(\underline{\qquad}, \mathsf{m})$

where the first argument of loves is missing. As Frege puts it, the verb phrase expresses something unsaturated, a function whose arguments are things that can fill the empty slot. In order to express this idea formally, we will make use of a device known as an ABSTRACTION OPERATOR. We will use a variable as a placeholder in the empty slot, and we will use an abstraction operator, written as the Greek letter λ ('lambda'), to bind that variable, creating a function that will accept a filler for that slot. This device is also known as LAMBDA ABSTRACTION.

The language of the SIMPLY TYPED LAMBDA CALCULUS, developed by the logician Alonzo Church, gives us the tools to represent 'unsaturated meanings' as functions. Using the λ symbol, we can ABSTRACT OVER the missing piece. The result looks like this:

(2) $\lambda x . \mathsf{loves}(x, \mathsf{m})$

This expression (read 'lambda x dot loves x m') denotes a function from an individual to a truth value, which yields true if and only if that individual loves Mary. This is the characteristic function of the set of all individuals that love Mary. In Chapter 2, we assumed that verb phrases (as well as nouns) denote sets of individuals. Apart from replacing sets by their characteristic functions, we are making the same assumption here.

A similar problem arises with expressions like *Everything*. A sentence containing *everything* is always translated as something of the following form:

$$\forall x.\underline{\qquad}(x)$$

where _____ is a placeholder for some predicate. For instance, *Everything is temporary* could be expressed:

(3) $\forall x.\mathsf{temporary}(x)$

while *Everything is permanent* would be expressed:

(4) $\forall x.\mathsf{permanent}(x)$

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:

(5) $\lambda P.\forall x.P(x)$

This 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.

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. It also means that expressions other than terms can serve as arguments to other expressions. So the logic in this chapter is different from $\mathrm{L}_{\mathrm{Pred}}$ 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

5.2.1 Types

Our languages L_{Prop}, L_0, and L_{Pred} had a rather limited set of syntactic categories: terms, predicates and functions 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:

(6) e

(the type of entities) for individual-denoting expressions (corresponding to TERMS in L_0), and

(7) 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 predicate logic will continue to be a formula.

From these types we will build up FUNCTION TYPES such as:

(8) $\langle e, t\rangle$

for expressions denoting functions from individuals to truth values. 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 $\langle\sigma,\tau\rangle$ 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 associates 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; $D_t = \{1,0\}$. 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*.

The system used in this book and in most semantic research 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 $[f(x)](y)$ de-

[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]."

notes 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 $[f(x)](y) = \mathsf{T}$ if and only if $\langle y, x\rangle \in R$.

For example, say we want to turn the following binary relation over the set of ABBA members into a function of type $\langle e, \langle e, t\rangle\rangle$:

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

The characteristic function of this relation, call it f, is shown below. Applied to any pair of individuals, it returns a truth value: T if that pair is in the relation, F otherwise.

(10)
$$f = \begin{bmatrix} \langle \text{Agnetha}, \text{Agnetha}\rangle & \to & \mathsf{F} \\ \langle \text{Agnetha}, \text{Benny}\rangle & \to & \mathsf{F} \\ \langle \text{Agnetha}, \text{Björn}\rangle & \to & \mathsf{F} \\ \langle \text{Agnetha}, \text{Frida}\rangle & \to & \mathsf{T} \\ \langle \text{Benny}, \text{Agnetha}\rangle & \to & \mathsf{F} \\ \langle \text{Benny}, \text{Benny}\rangle & \to & \mathsf{F} \\ \langle \text{Benny}, \text{Björn}\rangle & \to & \mathsf{F} \\ \langle \text{Benny}, \text{Frida}\rangle & \to & \mathsf{F} \\ \langle \text{Björn}, \text{Agnetha}\rangle & \to & \mathsf{F} \\ \langle \text{Björn}, \text{Benny}\rangle & \to & \mathsf{T} \\ \langle \text{Björn}, \text{Björn}\rangle & \to & \mathsf{T} \\ \langle \text{Björn}, \text{Frida}\rangle & \to & \mathsf{F} \\ \langle \text{Frida}, \text{Agnetha}\rangle & \to & \mathsf{F} \\ \langle \text{Frida}, \text{Benny}\rangle & \to & \mathsf{F} \\ \langle \text{Frida}, \text{Björn}\rangle & \to & \mathsf{T} \\ \langle \text{Frida}, \text{Frida}\rangle & \to & \mathsf{F} \end{bmatrix}$$

Left-to-right currying turns f into the function we call $f_{\to}$:

(11) $$f_{\rightarrow} = \begin{bmatrix} \text{Agnetha} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{F} \\ \text{Frida} & \rightarrow & \mathsf{T} \end{bmatrix} \\ \text{Benny} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{F} \\ \text{Frida} & \rightarrow & \mathsf{F} \end{bmatrix} \\ \text{Björn} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{T} \\ \text{Björn} & \rightarrow & \mathsf{T} \\ \text{Frida} & \rightarrow & \mathsf{F} \end{bmatrix} \\ \text{Frida} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{T} \\ \text{Frida} & \rightarrow & \mathsf{F} \end{bmatrix} \end{bmatrix}$$

Right-to-left currying turns f into the function we call $f_{\leftarrow}$:

(12) $$f_{\leftarrow} = \begin{bmatrix} \text{Agnetha} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{F} \\ \text{Frida} & \rightarrow & \mathsf{F} \end{bmatrix} \\ \text{Benny} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{T} \\ \text{Frida} & \rightarrow & \mathsf{F} \end{bmatrix} \\ \text{Björn} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{T} \\ \text{Frida} & \rightarrow & \mathsf{T} \end{bmatrix} \\ \text{Frida} & \rightarrow & \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{T} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{F} \\ \text{Frida} & \rightarrow & \mathsf{F} \end{bmatrix} \end{bmatrix}$$

Both $f_{\rightarrow}$ 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 turns 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 are two ordered pairs in the relation whose second element is Björn, namely, ⟨ Björn, Björn ⟩ and ⟨ Frida, Björn ⟩ (we look at the second element because the relation is right-to-left curried.) Accordingly, when we apply $f_{\leftarrow}$ to Björn, the result is a function that maps Björn to T Frida to T and the others to F

(13) $$f_{\leftarrow}(\text{Björn}) = \begin{bmatrix} \text{Agnetha} & \rightarrow & \mathsf{F} \\ \text{Benny} & \rightarrow & \mathsf{F} \\ \text{Björn} & \rightarrow & \mathsf{T} \\ \text{Frida} & \rightarrow & \mathsf{T} \end{bmatrix}$$

As another example, when $f_{\leftarrow}$ is applied to Agnetha, it returns a function that maps everything to F because there is no ordered pair in the relation whose second element is Agnetha. 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, *Björn*, in *Agnetha loves Björn*) to return a new function, which then may apply to the subject (say, *Agnetha*). That way, every part of the sentence is assigned a denotation, including the verb phrase (*loves Björn*), 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

(14) loves

where the initial lower case letter indicates that it is a function symbol rather than a relation symbol. Then

(15) $\mathsf{loves}(\mathsf{b})$

will serve as the translation for the verb phrase *loves Björn*, and

(16) $\mathsf{loves}(\mathsf{b})(\mathsf{a})$

will serve as the translation for *Agnetha loves Björn.* Note that in $\mathsf{loves}(\mathsf{b})(\mathsf{a})$, the subexpression $\mathsf{loves}(\mathsf{b})$ forms a unit. We have $\mathsf{loves}(\mathsf{b})(\mathsf{a})$ rather than $\mathsf{loves}(\mathsf{a})(\mathsf{b})$ because the verb combines first with the object *Björn* and then with the subject *Agnetha*. 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 $\mathsf{loves}(\mathsf{b})(\mathsf{a})$, we will write *as a shorthand*:

(17) $\mathsf{loves}(\mathsf{a}, \mathsf{b})$

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 (18a), with a two sets of parens, we will represent the denotation of a transitive verb in lambda calculus in the relational style as in (18b), with just one set of parens:

(18) a. $\lambda y . \lambda x . \mathsf{loves}(y)(x)$

b. $\lambda y \,.\, \lambda x \,.\, \mathsf{loves}(x, y)$

The expression in (18b) denotes the result of right-to-left currying the binary relation denoted by the binary predicate loves in predicate logic. Using the relational style 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 that style would be represented as '$\lambda y \,.\, \lambda x \,.\, x$ loves y', with a blend of English and lambda calculus.

5.2.2 Syntax and semantics

The introduction of an infinite set of syntactic categories sets the stage for the introduction of the LAMBDA OPERATOR (or λ-operator), also known as an ABSTRACTION OPERATOR. The lambda operator allows us to describe a wide range of functions. For example:

(19) $\lambda x \,.\, \mathsf{loves}(\mathsf{m}, x)$

denotes the characteristic function of the set of individuals that Mary loves, while

(20) $\lambda x \,.\, \mathsf{loves}(x, \mathsf{m})$

denotes the characteristic function of the set of individuals that love Mary. You can think of the λ-operator analogously to predicate notation for building sets. $\lambda x \,.\, \mathsf{loves}(\mathsf{m}, x)$ denotes the characteristic function of the set $\{x \mid \text{Mary loves } x\}$, that is, of the set of individuals that Mary loves. (It is common not to distinguish between sets and their characteristic functions. So we will often also say slightly imprecise things like "$\lambda x \,.\, \mathsf{loves}(\mathsf{m}, x)$ denotes the set of individuals that Mary 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 $\langle e, e\rangle$:

(21) $\lambda x.\mathsf{spouseOf}(\mathsf{loverOf}(x))$

This function takes as input an individual x and returns as output another individual, the spouse of x's lover.

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 happy, and F (False) if not. So $\lambda x.\mathsf{happy}(x)$ denotes the characteristic function of the set of happy individuals.

To give the full picture of the indirect interpretation theory, the syntactic constituent "loves Björn" will be 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"). The $[\![\cdot]\!]^{M,g}$ denotation function maps this lambda expression to the characteristic function of the set of individuals who love Björn in M. If that characteristic function is given the individual Agnetha as an input, the output is a truth value: T if Agnetha loves Björn 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.

5.2.3 Application and beta reduction

The functions resulting from abstraction behave just like the functions we are already familiar with. As in L_0, we indicate the arguments of a function using parentheses. This is called APPLICATION. If π is an expression denoting a function, and α is an expression whose type is the input type of π, then $\pi(\alpha)$ denotes the result of applying π to α, and its type is the output type of π. For example, $[\lambda x.\mathsf{happy}(x)](\mathsf{a})$ denotes the result of applying 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

(22) $[\lambda x.\mathsf{loves}(x,\mathsf{b})](\mathsf{a})$

denotes the result of applying the function 'loves Björn' to Agnetha.

Here is the syntax rule that introduces function application terms into the language:

Syntax Rule: Function 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: Function application**
> If α is an expression of type $\langle\sigma,\tau\rangle$, and β is an expression of type σ, then $[\![\alpha(\beta)]\!]^{M,g} = [\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$.

The expression we have just seen is provably equivalent to the simpler:

(23) $\textsf{loves}(\textsf{a},\textsf{b})$

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:

$$[\lambda x . \dots x \dots](\alpha)$$

can be simplified to

$$\dots \alpha \dots$$

The following pairs of expressions are equivalent; in each case, the second is the beta-reduced version of the first.

(24) a. $[\lambda x . \textsf{smiled}(x)](\textsf{a})$
b. $\textsf{smiled}(\textsf{a})$

(25) a. $[\lambda x . [\textsf{smiled}(x) \wedge \textsf{happy}(x)]](\textsf{a})$
b. $[\textsf{smiled}(\textsf{a}) \wedge \textsf{happy}(\textsf{a})]$

(26) a. $[\lambda x . [\textsf{smiled}(x) \wedge \textsf{happy}(y)]](\textsf{a})$
b. $[\textsf{smiled}(\textsf{a}) \wedge \textsf{happy}(y)]$

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

(27) $\lambda x.\mathsf{loves}(x, \mathsf{bj})$

is

(28) $\mathsf{loves}(x, \mathsf{bj})$.

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 be described as *taking the value description and replacing all free occurrences of the lambda-bound variable with the argument.* 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 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:

(29) 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

(30) $\lambda x.\mathsf{loves}(x, y)$

into

(31) $\lambda z.\mathsf{loves}(z, y)$

Here we replaced x with z, which we could do because z did not already occur free in the variable description in (30). 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:

(32) $[\lambda y.\lambda x.\mathsf{loves}(x, y)](x)$

If we just substitute x in for y, we get:

(33) $\lambda x.\mathsf{loves}(x, x)$

The rule of beta-reduction does not allow this, because it contains the proviso, "provided that α [the argument] does not contain any free variables that occur 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, (33) is not equivalent to the original expression (32). The expression (33) denotes the set of self-lovers, while the expression (32) 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 (32) and get the following:

(34) $[\lambda y.\lambda z.\mathsf{loves}(z, y)](x)$

This expression has exactly the same denotation as (32). If we substitute x for y by performing beta reduction on (34), then we get the right result: an expression that has the same denotation as (32) (and (34)), but that cannot be simplified any further.

(35) $\lambda z.\mathsf{loves}(z, x)$

Whereas our former attempt in (33) 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 (32), we recommend first using alpha conversion to 're-letter' the bound variable as in (34), before carrying out beta reduction as usual. In the Lambda Calculator, the software accompanying this book that can be used as a tool in solving the exercises, this re-lettering procedure is enforced as a matter of practice.[2]

[2] A third rule, ETA REDUCTION, allows us to rewrite a lambda term of the shape $\lambda x.[\phi(x)]$ as just ϕ, and vice versa. For example, this rule ensures that $\lambda x.\textsf{smiled}(x)$ and $\lambda y.\textsf{smiled}(y)$ are equivalent to each other and to $\textsf{smiled}$. Two other examples: $\lambda y[\lambda x.\textsf{loves}(y)(x)]$ is equivalent to $\lambda y.\textsf{loves}(y)$, which in turn is equivalent to $\textsf{loves}$; and $\lambda P.[\lambda x.\neg P(x)]$ is equivalent to $\lambda P.\neg P$. 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, $\lambda x.\textsf{loves}(x)(x)$ (which denotes the set of self-lovers) cannot be eta-reduced to $\textsf{loves}(x)$, which denotes the set of x-lovers rather than self-lovers. If it is not clear why $\textsf{loves}(x)$ has this denotation, it might help to see that it can be obtained via eta reduction from $\lambda y.\textsf{loves}(x)(y)$, 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 $\lambda x.[(\lambda P.P)(\textsf{smiled})](x)$ and $\textsf{smiled}$ 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 term. 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.

5.2.4 Some 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:

(36) $\lambda x.\mathsf{smokes}(x)$

Then *non-smoker* should denote the set of people who don't smoke, and it should translate to this:

(37) $\lambda x.\neg\mathsf{smokes}(x)$

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:

(38) $\lambda P.[\lambda x.\neg P(x)]$

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:

(39) 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:

(40) $\forall P.[[\exists x.\mathsf{good}(x) \wedge \mathsf{general}(x) \wedge P(x)] \rightarrow P(\mathsf{napoleon})]$

(41) $\neg[\exists x \exists y.[\neg(x = y) \wedge [\forall P.P(x) \leftrightarrow P(y)]]]$

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 3, all variables range over individuals; in other words, the types of all first-order variables is e.

Other higher-order expressions that we can treat using our new language include quantifiers like *every cellist* and determiners like *every*. 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:

(42) $\lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x)$

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 suggests that *cellist* is translated as follows:

(43) $\lambda y.\mathsf{cellist}(y)$

Then *every cellist* will be translated as:

(44)
$$[\lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x)](\lambda y.\mathsf{cellist}(y))$$
$$\begin{array}{cc} \lambda P.\lambda Q.\forall x.P(x) \rightarrow Q(x) & \lambda y.\mathsf{cellist}(y) \\ | & | \\ \textit{every} & \textit{cellist} \end{array}$$

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:

(45) $\lambda Q.\forall x.[\lambda y.\mathsf{cellist}(y)](x) \rightarrow Q(x)$

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:

(46) $\lambda Q.\forall x.\mathsf{cellist}(x) \rightarrow Q(x)$

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:

(47) $[\lambda Q.\forall x.\mathsf{cellist}(x) \rightarrow Q(x)](\lambda z.\mathsf{smokes}(z))$

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:

(48) $[\forall x.\mathsf{cellist}(x) \rightarrow [\lambda z.\mathsf{smokes}(z)](x)]$

In a second step, we apply $\lambda z.\mathsf{smokes}(z)$ to x and get $\mathsf{smokes}(x)$. So we end up with:

(49) $\forall x.\mathsf{cellist}(x) \rightarrow \mathsf{smokes}(x)$

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.

5.3 Summary

5.3.1 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. This means that all those expressions that were already well-formed formulas in our old logics $\mathrm{L}_{\mathrm{Prop}}$ and L_0 (e.g. atomic formulas, conjunctions, disjunctions, quantified statements, etc.) are expressions of type t.

For every type, there is a set of constants of that type, and an infinite set of variables of that type. Each variable bears an index, indicated with a subscripted integer.

1. **Basic Expressions**
   For every type τ, there is:
   - a possibly empty set of constants Con_τ

- an infinite set of variables Var_τ, each bearing a natural number as an index, one for each natural number. (The index 0 can be suppressed, so x is an abbreviation of x_0. Abbreviated and non-abbreviated forms should not occur in the same formula, lest confusion arise.)

In this language, since we can have constants of any type, including the whole range of functional types, we will drop the convention that constants denoting functions end in Of. For instance, we could have a constant non $\in \text{Con}_{\langle\langle e,t\rangle,\langle e,t\rangle\rangle}$ that denotes the function we associated with the English prefix *non-* above. The constant loves above is in $\text{Con}_{\langle e,\langle e,t\rangle\rangle}$.

Variables of the form x_i, y_i or z_i, where i is an integer, are variables of type e. Variables of the form P_i or Q_i are of type $\langle e,t\rangle$. Variables of the form R_i are of type $\langle e,\langle e,t\rangle\rangle$. Outside of these conventions, we sometimes indicate the type of a variable by means of an additional subscript.

2. **Application** (cf. 'Complex Terms' in L_{Pred})
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 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 $[\phi \leftrightarrow \psi]$.

(This means that we cannot apply the connectives to expressions of any type other than t, nor can we use them to produce any such expressions.)

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** (new!)
   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. $\lambda x \lambda y . \mathsf{Admires}(x, y)$. (The order of the lambdas still matters: this function is not the same as $\lambda y \lambda x . \mathsf{Admires}(x, y)$.) 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. $\lambda x \lambda y . \exists z . \mathsf{Gave}(x, y, z)$. 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,

(50) $\mathsf{loves}(x, y)$

is defined to be equivalent to $\mathsf{loves}(y)(x)$. In general, if π denotes an n-place right-to-left curried relation, then

(51) $\pi(\alpha_1)(\alpha_2)...(\alpha_n)$

can be re-written as

(52) $\pi(\alpha_n, \alpha_{n-1}, ..., \alpha_1)$

Exercise 3. Consider the following expressions, assuming the following abbreviations:

- x is $v_{0,e}$ (meaning that x is variable number 0 of type e)
- y is $v_{1,e}$
- P is $v_{0,\langle e,t\rangle}$, Q is $v_{1,\langle e,t\rangle}$, and X is $v_{2,\langle e,t\rangle}$
- R is $v_{0,\langle e\times e,t\rangle}$
- a is $c_{0,e}$ and b is $c_{1,e}$

1. $[\lambda x . P(x)](\mathsf{a})$
2. $[\lambda x . P(x)(\mathsf{a})]$
3. $[\lambda x . R(y, \mathsf{a})]$
4. $[\lambda x . R(y, \mathsf{a})](\mathsf{b})$
5. $[\lambda x . R(x, \mathsf{a})](\mathsf{b})$
6. $[\lambda x \lambda y . R(x, y)](\mathsf{b})$
7. $[\lambda x \lambda y . R(x, y)](\mathsf{b})(\mathsf{a})$
8. $[\lambda x . [\lambda y . R(x, y)](\mathsf{b})](\mathsf{a})$
9. $[\lambda X . \exists x . [P(x) \wedge X(x)]](\lambda y . R(\mathsf{a}, y))$
10. $[\lambda X . \exists x . [P(x) \wedge X(x)]](\lambda x . R(\mathsf{a}, x))$
11. $[\lambda X . \exists x . [P(x) \wedge X(x)]](\lambda y . R(y, x))$

12. $[\lambda X\,.\,\exists x\,.\,[P(x) \wedge X(x)]](Q)$

13. $[\lambda X\,.\,\exists x\,.\,[P(x) \wedge X(x)]](X)$

14. $[\lambda X\,.\,\exists x\,.\,[P(x) \wedge X(x)](\lambda x\,.\,Q(x))]$

15. $[\lambda y \lambda x\,.\,R(y,x)](\mathsf{a})$

For each of the above, answer the following questions:

(a) Is it a well-formed expression of L_λ (given both the official syntax and our abbreviatory conventions) and if yes, what is its type?

(b) If the formula is well-formed, give a completely beta-reduced expression which is equivalent to it. Use alpha-conversion (re-lettering of bound variables) if necessary to avoid variable clash.

You can check your answers using the Lambda Calculator.

Exercise 4. Identify the type of each of the following. Assume that man and mortal are constants of type $\langle e,t\rangle$.

1. $\lambda y\,.\,y$
2. $\lambda x\,.\,P(x)$
3. P
4. a
5. x
6. $P(x)$

7. $[\lambda x.P(x)](\mathsf{a})$
8. $P(\mathsf{a})$
9. $R(x, y)$
10. $\lambda x.R(x, \mathsf{a})$
11. $\lambda y \lambda x.R(y, x)$
12. $[\lambda y \lambda x.R(y, x)](\mathsf{a})$
13. $[\lambda x.R(y, \mathsf{a})](\mathsf{b})$
14. $R(\mathsf{a}, \mathsf{b})$
15. $\lambda x.[P(x) \wedge Q(x)]$
16. $[\lambda x.P(x) \wedge Q(x)](\mathsf{a})$
17. $\lambda x \lambda y.[R(y)(\mathsf{a}) \wedge Q(x)]$
18. $\lambda P.P$
19. $\lambda P.P(\mathsf{a})$
20. $\exists x.P(x)$
21. $\lambda P.\exists x.P(x)$
22. $[\lambda P.\exists x.P(x)](\mathsf{man})$
23. $\exists x.\mathsf{man}(x)$
24. $\lambda P.\forall x.P(x)$
25. $[\lambda P.\forall x.P(x)](\mathsf{mortal})$
26. $\neg\mathsf{mortal}(x)$

27. $\lambda x.\neg\mathsf{mortal}(x)$
28. $\lambda P\lambda x.\neg P(x)$
29. $[\lambda P\lambda x.\neg P(x)](\mathsf{mortal})$
30. $\lambda x.\neg\mathsf{mortal}(\mathsf{a})$
31. $[\lambda x.\neg\mathsf{mortal}(x)](\mathsf{a})$
32. $\neg\mathsf{mortal}(\mathsf{a})$
33. $\lambda Q.\forall x.[\mathsf{man}(x)\rightarrow Q(x)]$
34. $[\lambda Q.\forall x.[\mathsf{man}(x)\rightarrow Q(x)]](\mathsf{mortal})$
35. $\lambda P\lambda Q.\forall x.[P(x)\rightarrow Q(x)]$
36. $[\lambda P\lambda Q.\forall x[P(x)\rightarrow Q(x)]](\mathsf{man})$
37. $[\lambda P\lambda Q.\forall x[P(x)\rightarrow Q(x)]](\mathsf{man})(\mathsf{mortal})$
38. $[\lambda Q.\forall x[\mathsf{man}(x)\rightarrow Q(x)]](\lambda x[\mathsf{mortal}(x)])$
39. $[\lambda P\lambda x.\neg P(x)](\mathsf{mortal})$
40. $[\lambda P\lambda x.\neg P(x)](\lambda x.\mathsf{mortal}(x))$

You can check your answers using the Lambda Calculator.

Exercise 5. Where possible, apply beta-reduction to give a more concise version of each of the following. If the expression is fully reduced, just give the original expression.

1. $[\lambda x.x](\mathsf{a})$
2. $[\lambda P.P](\mathsf{man})$

3. $[\lambda x . P(x)](\text{a})$
4. $[\lambda x . P(x)]$
5. $[\lambda y \lambda x . R(y, x)](\text{a})$
6. $[\lambda x . R(y, \text{a})](\text{b})$
7. $[\lambda P . \exists x . P(x)](\text{man})$
8. $[\lambda P . \forall x . P(x)](\text{mortal})$
9. $\lambda x . \neg\text{mortal}(x)$
10. $[\lambda P \lambda x . \neg P(x)](\text{mortal})$
11. $[\lambda x . \neg\text{mortal}(x)](\text{a})$
12. $[\lambda Q . \forall x . [\text{man}(x) \rightarrow Q(x)]](\text{mortal})$
13. $[\lambda P \lambda Q . \forall x . [P(x) \rightarrow Q(x)]](\text{man})$
14. $[\lambda P \lambda Q . \forall x . [P(x) \rightarrow Q(x)]](\text{man})(\text{mortal})$
15. $[\lambda x . P(x) \wedge Q(x)](\text{a})$
16. $[\lambda x \lambda y . [R(y, \text{a}) \wedge Q(x)]](\text{a})(\text{b})$
17. $[\lambda x . \exists y . R(x, y)](y)$
18. $[\lambda x . \text{a}](\text{b})$
19. $[\lambda x . [P(x) \rightarrow \exists x . R(\text{b}, x)]](\text{a})$
20. $[\lambda Q . \forall x[\text{mortal}(x) \rightarrow Q(x)]](\lambda x[\text{mortal}(x)])$
21. $[\lambda Q . \exists P . \forall x . [P(x) \rightarrow Q(x)]](\text{mortal})$
22. $[\lambda P \lambda x . \neg P(x)](\lambda x[\text{mortal}(x)])$
23. $[\lambda P \lambda x . P(x)](\lambda x[\neg\text{mortal}(x)])$

You can check your answers using the Lambda Calculator.

5.3.2 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 $\{\mathsf{T},\mathsf{F}\}$.

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:[3]

- $(D_\tau)_{\tau\in\mathcal{T}}$ 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.

So much for models; now to assignments. 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:

[3] This formalization is inspired by Gallin (1975), p. 12.

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 $[\![\psi]\!]^{M,g} = \mathsf{T}$.

   (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) 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, $f(o) = [\![\alpha]\!]^{M,g[u \mapsto o]}$.

Exercise 6.

(a) Partially define a model for L_λ giving denotations to the constants loves, n, and d of type $\langle e, \langle e, t\rangle\rangle$, e, and e, respectively.

(b) Show that $[\lambda x.\mathsf{loves}(\mathsf{n})(x)](\mathsf{d})$ and its beta-reduced version $\mathsf{loves}(\mathsf{n})(\mathsf{d})$ have the same semantic value in your model using the semantic rules for L_λ.

Exercise 7. 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 8. 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 term 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 9. 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.)

5.4 Further reading

This chapter has provided just the bare minimum that is needed for starting to do formal semantics. There is no trace of 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.

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 terms 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. The process by which translations of complex expressions are derived compositionally 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 *Agnetha* will translate as the type e expression a; both denote the individual Agnetha. 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. *Agnetha* $\rightsquigarrow \mathsf{a}$
b. *smiled* $\rightsquigarrow \lambda x\,.\,\mathsf{smiled}(x)$

The combination, *Agnetha smiled*, will then be translated as the result of applying the translation of the verb to the translation of the subject:

(2)
$$\Tree [.{[\lambda x\,.\,\mathsf{smiled}(x)](\mathsf{a})} [.{\mathsf{a}} \textit{Agnetha}] [.{\lambda x\,.\,\mathsf{smiled}(x)} \textit{smiled}]]$$

... or equivalently, through beta reduction:

(3)
$$\Tree [.{\mathsf{smiled}(\mathsf{a})} [.{\mathsf{a}} \textit{Agnetha}] [.{\lambda x\,.\,\mathsf{smiled}(x)} \textit{smiled}]]$$

The formulas at the tops of these trees have the same denotation as each other and as the English sentence *Agnetha 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) $[\![\textit{Agnetha smiled}]\!] = \mathsf{T}$

we instead write:

(5) *Agnetha smiled* $\rightsquigarrow$ $\mathsf{smiled}(\mathsf{a})$

and:

(6) $[\![\mathsf{smiled}(\mathsf{a})]\!] = \mathsf{T}$

in order to express that the sentence is true (ignoring here the usual adornment of the denotation brackets with a specification of a model and an assignment function). 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 "Agnetha" is not itself the person Agnetha. 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 grammar', 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.

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
S	→	S CoordP
CoordP	→	Coord S
VP	→	V (DP|AP|PP|NegP)
NegP	→	Neg VP|AP
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:

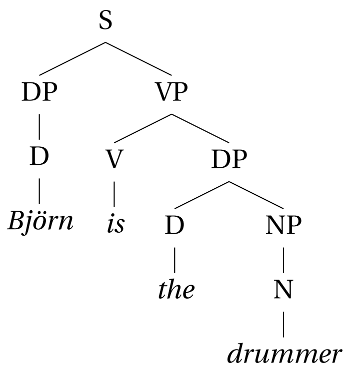
(8) **Lexicon**

Coord: *and*, *or*
Neg: *not*

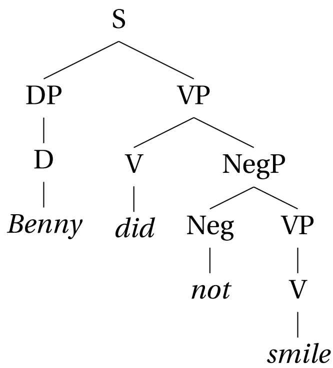
V: *smiled, laughed, loves, hugged, is, did*
A: *Swedish, happy, kind, proud*
N: *singer, drummer, musician*
D: *the, a, every, some, no*
D: *Agnetha, Frida, Björn, Benny, everybody, somebody, nobody*
P: *of, with*

For example, this grammar generates *Björn is the drummer* and *Benny did not smile*, with syntactic structures as shown in the following analysis trees:

(9)



(10)



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.

(a) George loves everybody.

(b) Some drummer smiled every happy musician.

(c) Agnetha is not a drummer.

(d) Frida is.

(e) No is a happy singer.

(f) Somebody is proud of the singer.

(g) A drummer loves proud of Björn.

(h) The proud drummer of Björn loves every happy happy happy happy drummer.

(i) Frida smiles with nobody.

(j) Agnetha and Frida are with Björn.

(k) Agnetha is with Björn and Frida is with Benny.

Keep in mind that the syntax might generate sentences that don't make any sense, 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:

(11)
DP
|
Agnetha

instead of

(12)
$$\begin{array}{c}\text{DP}\\ |\\ \text{D}\\ |\\ \textit{Agnetha}\end{array}$$

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 1. 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.

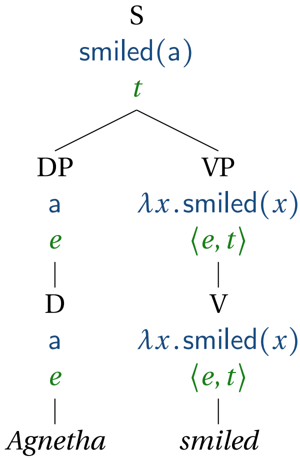
then what type does the translation of γ have, assuming that it is translated according to the rule of Function Application?

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 2. 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 *Agnetha smiled* as follows (showing only fully beta-reduced translations at each node, along with their types):

(13)



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. Theories such as Construction Grammar (Croft & Cruse, 2004) 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.

6.2 Fun with Function Application

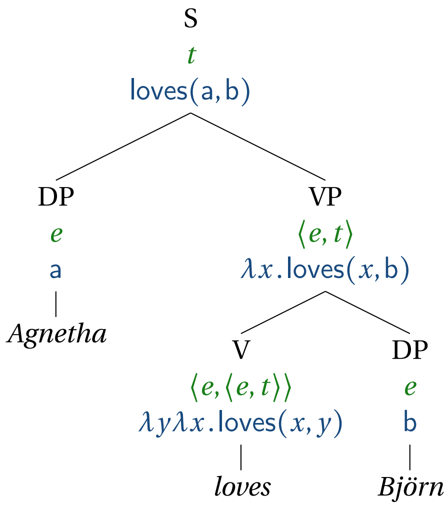
6.2.1 *Agnetha loves Björn*

Let us now consider how to analyze a simple transitive sentence like *Agnetha loves Björn*. We will represent the denotation of the verb *loves* as follows:

(14) *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 *Agnetha loves Björn*:

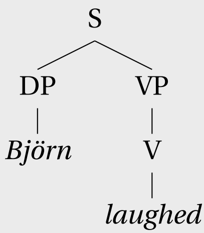
(15)



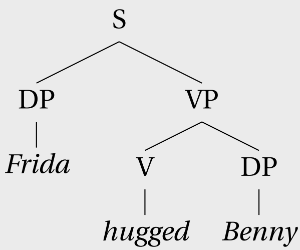
Via Function Application, the transitive verb *loves* combines with the object *Björn*. The VP *loves Björn* thus comes to denote (the characteristic function of) the set of all individuals who love Björn, which we can think of as the property of loving Björn. This property is then attributed to Agnetha through a second application of Function Application at the top node.

Exercise 3. For both of the following trees, give a fully beta-reduced translation at each node. Give appropriate lexical entries for words that have not been defined above.

(a)



(b)



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 tree in Figure 6.1. You will also need to assume a lexical entry for *to* that works along with your assumption about *introduce* and the structure of the syntax tree.

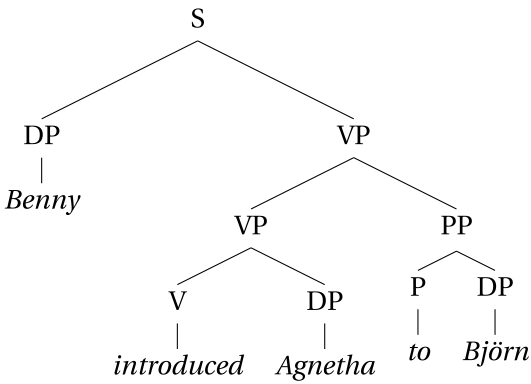
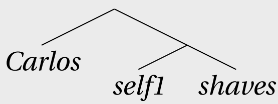


Figure 6.1: A ditransitive verb

Exercise 5. 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.

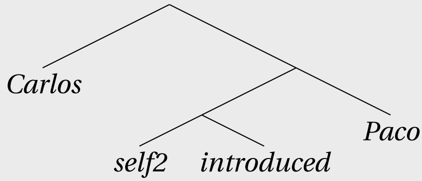
Assume that *Carlos* is an ordinary proper name, translated as a constant of type e, and assume that *shaves* is an ordinary transitive verb, translated as an expression of type $\langle e, \langle e, t\rangle\rangle$. You can use the lexical entry for *introduce* given in the previous exercise.

(a) For the sentence *Carlos self1-shaves*, make the structure below yield the denotation 'Carlos shaves himself' by supplying the denotation of *self1*.

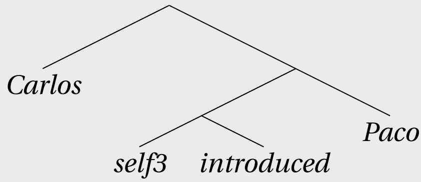


(b) For the sentence *Carlos self2-introduced Paco*, make the structure below yield the denotation 'Carlos introduced Paco to

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*.



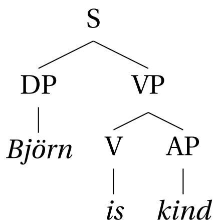
Make sure that your denotations work not just for sentences involving Carlos and Paco, but arbitrary proper names.

(Exercise due to Maribel Romero.)

6.2.2 *Björn is kind*

Now let us consider how to analyze a sentence with an adjective following *is*, such as *Björn is kind*. The syntactic structure is as follows:

(16)



We will continue to assume that the proper name *Björn* 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.

(17) *kind* $\rightsquigarrow \lambda x\,.\,\mathsf{kind}(x)$

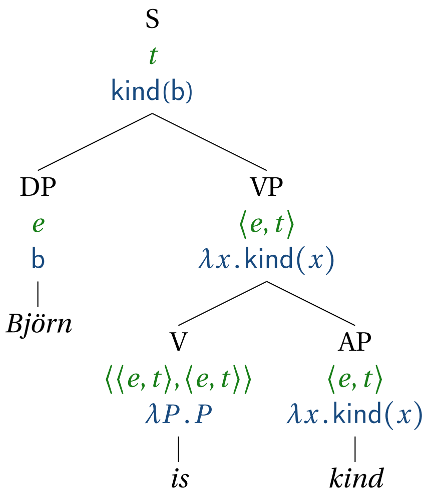
Now, what is the contribution of the copula *is*? Besides signaling present tense, it does not seem to accomplish more than to link the predicate 'kind' with the subject of the sentence. Since we have not started dealing with tense yet, we will ignore the former function and focus on the latter. (We 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 *Benny did not smile*.)

(18) *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 *Björn is kind*:

(19)



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 (*Björn*, *is*, and *kind*).

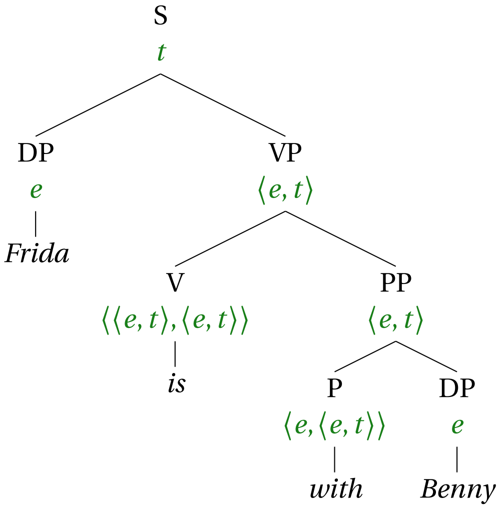
6.2.3 *Frida is with Benny*

Like adjectives, prepositional phrases can also serve as predicates, as in, for example, *Frida is with Benny*. Let us translate *with* as follows, invoking a binary predicate with:

(20) *with* $\rightsquigarrow \lambda y \lambda x . \mathsf{with}(x, y)$

Via Function Application, the preposition *with* combines with its object *Benny*, 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 *Frida* to produce a truth value.

(21)



Exercise 6. Derive the translation into L_λ for *Frida is with Benny* by giving a fully beta-reduced translation for each node.

6.2.4 *Benny is proud of Frida*

Like prepositions, adjectives can denote functions of type $\langle e,\langle e,t\rangle\rangle$. *proud* is an example; in *Benny is proud of Frida*, the adjective *proud* expresses a relation that holds between Benny and Frida. We can capture this by assuming that *proud* translates as:

$$\lambda y \lambda x\,.\,\mathsf{proud}(x, y)$$

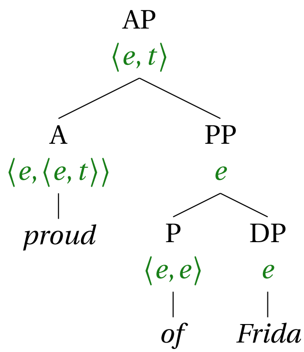
This is an expression of type $\langle e,\langle e,t\rangle\rangle$ denoting a function that takes two arguments, first a potential object of pride (such as Frida), then a potential bearer of such pride (e.g. Benny), and returns True if the pride relation holds between them.

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 $\langle e, e \rangle$.

(22) *of* $\rightsquigarrow \lambda x \,.\, x$

So the adjective phrase *proud of Frida* will have the following structure:

(23)



Exercise 7. Give a lexical entry for *proud* and a fully beta-reduced form of the translation at each node for *Benny is proud of Frida*. (You will need to draw out more of the tree structure than what is shown above.)

6.2.5 *Agnetha is a singer*

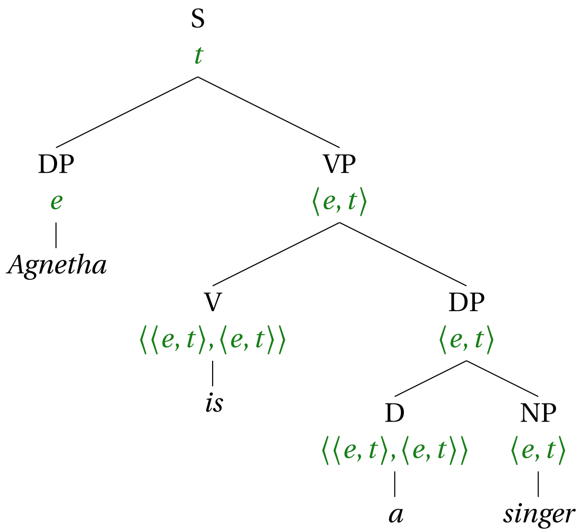
Let us consider *Agnetha is a singer*. The noun *singer* can be analyzed as an $\langle e, t \rangle$ type property like *Swedish*, the characteristic function of the set of individuals who are singers.

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.

(24) $a \rightsquigarrow \lambda P \,.\, P$

With these assumptions, the derivation will go as follows.

(25)

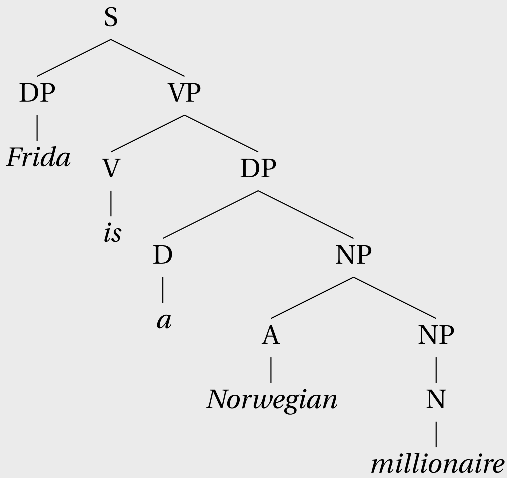


Exercise 8. Give fully beta-reduced translations at each node of the tree for *Agnetha is a singer*.

Exercise 9. Can we treat *a* as $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ in a sentence like *A singer loves Björn*? 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?

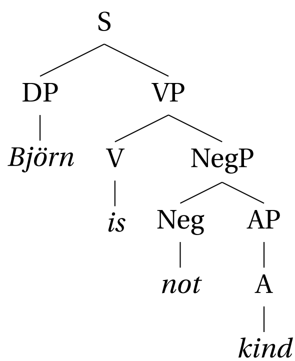
(26)



6.3 Negation

Now let us consider how to analyze the word *not* in a sentence like *Björn is not kind.* The syntactic structure would be as follows:

(27)



The denotation of *Björn is not kind* should be the negation of *Björn is kind*:

(28) $\neg\mathsf{kind}(\mathsf{b})$

Thus the property that *(is) not kind* denotes should be something that applies to an individual and yields ‘true’ only in case the that the individual is not kind:

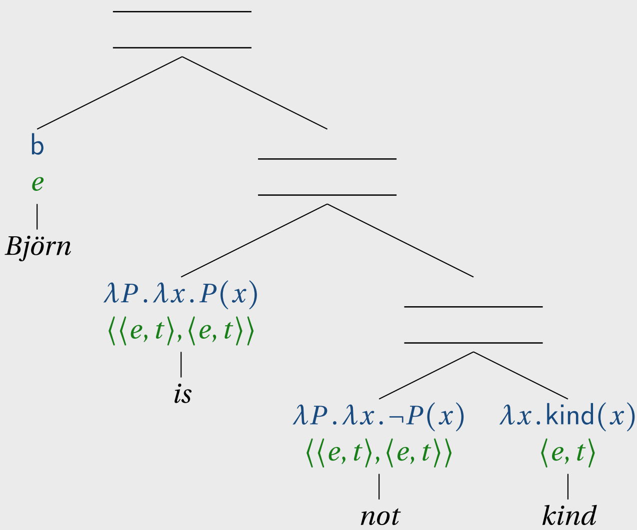
(29) $\lambda x.\neg\mathsf{kind}(x)$

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:

(30) *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 11. Using predicate negation, give a compositional analysis of *Björn is not kind*, by showing the translations and types at each node of the syntax tree.

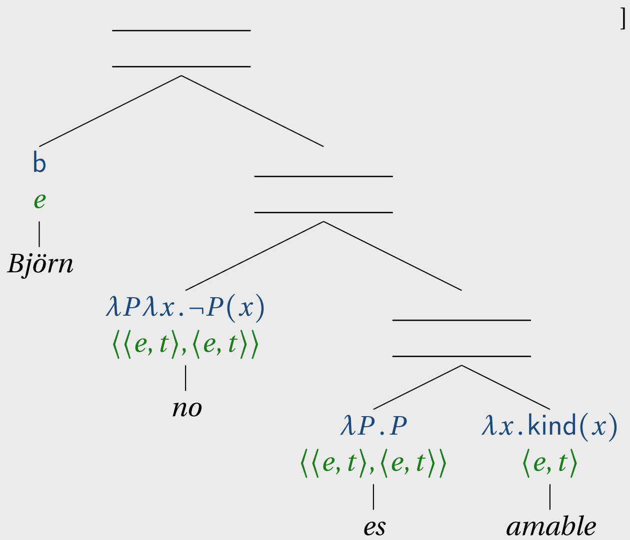


Exercise 12. Can the 'predicate negation' meaning for *not* provided in (30) also be applied to the negation of predicate nominals as in *Björn is not a singer*? Justify your answer.

Exercise 13. In Spanish, the word order and syntactic structure of negative sentences is slightly different than that of English, as shown below.

(31) Björn no es amable
Björn not is kind
'Björn is not kind'

Does the lexical entry provided in (30) give equivalent truth conditions in both Spanish and English despite this difference in word order? To answer this question, show the translations and types at each node of the syntax tree below.



6.4 Quantifiers: type $\langle\langle e,t\rangle,t\rangle$

6.4.1 Quantifiers

Let us now consider how to analyze quantifiers like *everybody* and *nobody*. Consider the sentence:

(32) Everybody smiled.

We have assumed that a VP like *smiled* denotes a predicate (type $\langle e,t\rangle$) and that a sentence like (32) denotes a truth value (type t). The translation of *Everybody smiled* should be something like the following (assuming that every individual in the domain is conceived of as human):

(33) *Everybody smiled* $\rightsquigarrow \forall x\,.\,\textsf{smiled}(x)$

Informally, then, the contribution of *everybody* to the denotation of a sentence is a template:

(34) $\forall x\,.\,\_(x)$

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":

(35) *everybody* $\rightsquigarrow \lambda P\,.\,\forall x\,.\,P(x)$

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:

(36) $\langle\langle e,t\rangle,t\rangle$

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:

(37)

$$\begin{array}{ccc} & \forall x.\textsf{smiled}(x) & \\ & t & \\ \lambda P.\forall x.P(x) & & \lambda x.\textsf{smiled}(x) \\ \langle\langle e,t\rangle,t\rangle & & \langle e,t\rangle \\ | & & | \\ \textit{everybody} & & \textit{smiled} \end{array}$$

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:

(38) *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:

(39) *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*.)

6.4.2 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:

(40) $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$

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:

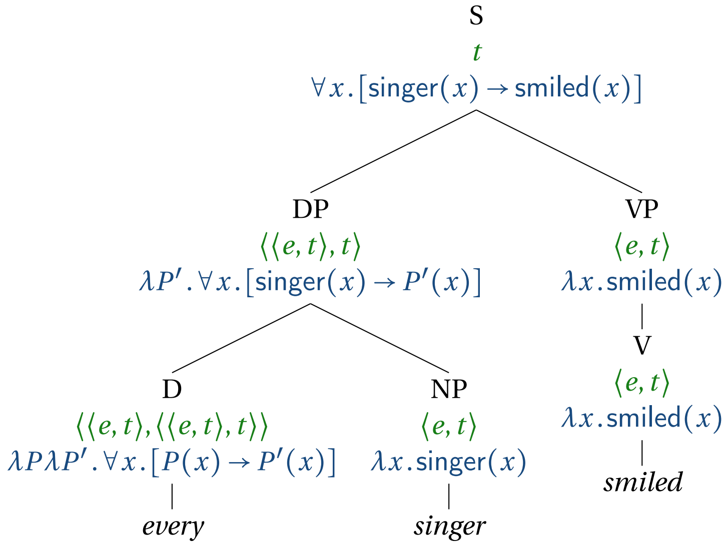
(41) *some* $\rightsquigarrow \lambda P \lambda P' . \exists x . [P(x) \wedge P'(x)]$

(42) *no* $\rightsquigarrow \lambda P \lambda P' . \neg\exists x . [P(x) \wedge P'(x)]$

(43) *every* $\rightsquigarrow \lambda P \lambda P' . \forall x . [P(x) \rightarrow P'(x)]$

These lexical entries will yield analyses like the following:

(44)

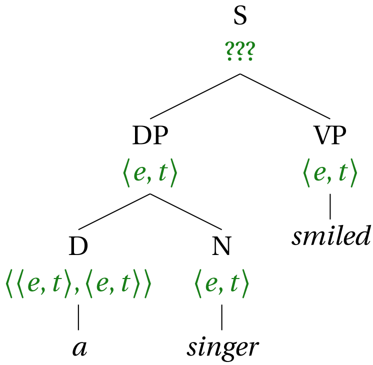


The same strategy can be applied to indefinite descriptions like *a singer*. Previously, we analyzed indefinite descriptions in sentences like *Agnetha is a singer*, where the indefinite description functions as a PREDICATE, and applies to a subject. But an indefinite description can also function as the subject or object of a transitive verb, as in the following sentences:

(45) a. A singer loves Frida. [subject position]
b. Frida 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:

(46)



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.

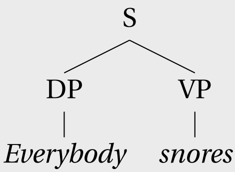
Exercise 14. Give an analysis of *A singer loves Frida* using Function Application and Non-Branching Nodes. Your analysis should take the form of a tree, specifying at each node, the syntactic category, the semantic type, and a fully beta-reduced translation to L_λ. The translation of the sentence should be true in any model where there is some individual that is both a singer and someone who loves Frida. 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 Frida, and no others.

Exercise 15. For each of the following trees, give the semantic type and a completely beta-reduced translation at each node. Give appropriate lexical entries for words that have not been defined above, following the style we have developed:

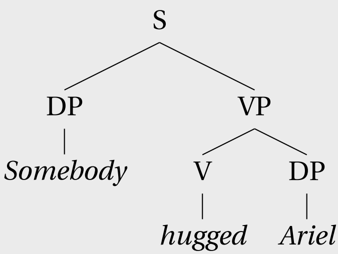
- Adjectives, non-relational common nouns, and intransitive verbs are of type $\langle e, t\rangle$.
- Transitive verbs are of type $\langle e, \langle e, t\rangle\rangle$.
- Proper names are of type e.
- Quantificational DPs are of type $\langle\langle e, t\rangle, t\rangle$.
- Determiners are of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$.

The lexical entries should be assigned in a way that captures what a model should be like if the sentence is true. For example, *Nobody likes Ursula* should be predicted to be true in a model such that no individual stands in the 'like' relation to Ursula.

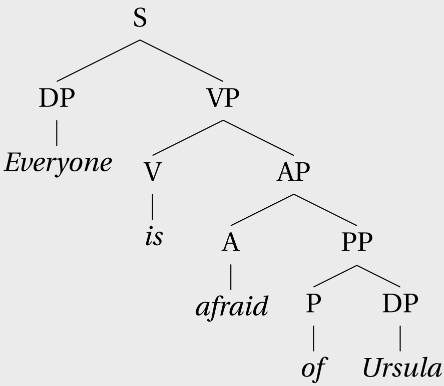
(a)



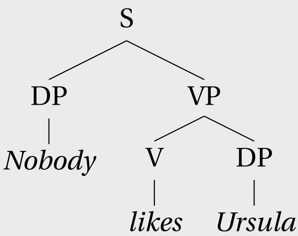
(b)



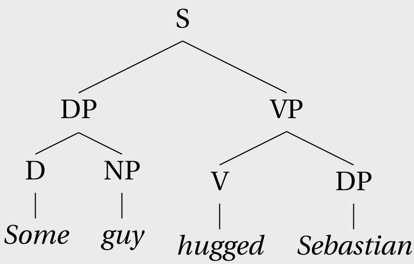
(c)



(d)



(e)



Exercise 16. In the early 70's, cases of VP coordination as in *Sam smokes or drinks* were analyzed using CONJUNCTION REDUCTION, a transformational rule that deletes the subject of the second clause under identity with the subject in the first clause, so this

sentence would underlyingly be *Sam smokes or Sam drinks*.

1. What translation into L_λ would the conjunction reduction analysis predict for a case like *Everybody smokes or drinks*?
2. What is problematic about this translation?
3. Give an alternative lexical entry for *or* that avoids the problem with the conjunction reduction analysis.
4. Give a syntax tree and a step-by-step derivation of the truth conditions for *Sam smokes or drinks* using your analysis.
5. Explain how your analysis avoids the problem.

6.4.3 Empirical diagnostics against type e

Under the analysis we have just given, quantifiers like *everybody* and *every singer* are treated not as type e but as type $\langle\langle e,t\rangle,t\rangle$. Is there any viable analysis on which they are type e instead? In this section, we list empirical diagnostics that can be used to argue against such an analysis. These diagnostics can be used to *show* that such expressions cannot be of type e. An expression of type e denotes a particular individual, so two occurrences of the same expression of type e denote the same individual (unless these expressions are context sensitive and the context changes from use to use—pronouns like *him* and *her*, for example which we will argue get their meaning from assignment functions just like variables in logic, are arguably type e and yet may refer to different individuals on different occasions of use). It follows that expressions of type e should exhibit certain properties.

An expression of type e should validate **subset-to-superset inferences**. For example:

(47) Susan came yesterday morning.
$\therefore$ Susan came yesterday.

This is correctly predicted to be a valid inference under the assumption that the subject (*Susan*) denotes an individual. Here is why. The set of things that came yesterday morning is a subset of the things that came yesterday. For any expression α (including *Susan*), if α denotes an individual, then α *came yesterday morning* is true if the individual denoted by α is among the things that came yesterday morning. But if that is true, then that individual is among the things that came yesterday. Hence if the first sentence is true, then the second sentence is true.

In contrast, the expression *at most one letter* fails to validate subset-to-superset inferences.

(48) At most one letter came yesterday morning.
$\therefore$ At most one letter came yesterday.

This inference is *not* valid, so *at most one letter* must not denote an individual, so it must not be of type e.

Exercise 17. Among the quantificational determiners *some, every, no, at least one, at most one,* which validate subset-to-superset inferences? Give examples.

A second property that expressions of type e have is related to the LAW OF EXCLUDED MIDDLE, which is a theorem of propositional and predicate logic. The law of excluded middle says that $[p \vee \neg p]$ is true for any formula p. That is to say, the disjunction of any p with its negation is a tautology. For example:

(49) John is over 30 years old, or John is not over 30 years old.

This is a tautology, and that is because *John* is an expression of type e. Any expression of type e will yield a tautology in a sentence like this. Here is why. Everything is either over 30 years old or not over 30 years old; together these two sets cover the entire set of individuals. If α is of type e, then α *is over 30 years old* is true iff the individual that α denotes is over 30 years old. α *is not over 30 years old* is true iff the individual is not over 30 years old. Since everything satisfies at least one of these criteria, sentence (49) (under a standard analysis of *or* as logical disjunction) cannot fail to be true.

But the following sentence is not a tautology (here, the second disjunct should be read with *everybody* taking scope over *not*):

(50) Every woman in this room is over 30 years old, or every woman in this room is not over 30 years old.

So *every woman* cannot be of type e.

Exercise 18. Among the quantificational determiners *some, every, no, at least one, at most one,* which give rise to tautologies in examples analogous to (49) and (50)? Give examples. For which of these cases, then, does this diagnostic provide evidence *against* a type e analysis?

A third property that expressions of type e should have is related to the LAW OF NON-CONTRADICTION, another theorem of propositional and predicate logic. The law of non-contradiction is the principle that $[p \wedge \neg p]$ is false for any formula p. That is to say, the conjunction of any p with its negation is a contradiction. If we take the sentence *Mont Blanc is higher than 4,000m* and conjoin it with its negation, the result is self-contradictory:

(51) Mont Blanc is higher than 4,000m, and Mont Blanc is not higher than 4,000m.

This sentence is self-contradictory *because* Mont Blanc denotes an individual. Here is why. Nothing that counts as 'higher than

4,000m' counts as 'not higher than 4,000m'; these two sets are disjoint. If α is of type e, then α *is higher than 4,000m* is true if and only if the individual that α denotes is higher than 4,000m. In that case, the second conjunct must be false. The same reasoning works in reverse; if the second conjunct is true, then the first must be false. The two conjuncts stand in contradictory opposition to each other, as p and $\neg p$ do. Hence, the conjunction (under an analysis of *and* as logical conjunction) can never be true.

The following sentence, however, is not self-contradictory:

(52) More than two mountains are higher than 4,000m, and more than two mountains are not higher than 4,000m.

Evidently, the two conjuncts do not stand in contradictory opposition to each other, and the law of contradiction does not prevent them from being true at the same time. If *more than two mountains* had type e, and picked out a particular individual, then we would expect the sentence to be self-contradictory. It is not, so *more than two mountains* must not have type e.

Importantly, it can happen that a given expression is not of type e, and yet still gives rise to a contradiction in sentences like this. For example:

(53) Every mountain is higher than 4,000m, and every mountain is not higher than 4,000.

This sentence is contradictory, but that is not grounds for concluding that *every mountain* is type e. The implication only goes in one direction: If a given expression fails to give rise to a contradiction in this type of example, then that is positive evidence that it is not type e (as long as it is not context-sensitive). If it gives rise to a contradiction, then it may or may not be type e.

Exercise 19. Among the quantificational determiners *some, every, no, at least one, at most one,* which give rise to contradictions in sentences like (51) and (52)? Give examples. For which of these cases, then, does this diagnostic provide evidence *against* a type e analysis?

Exercise 20. This sentence is not contradictory: *At most two mountains are higher than 4,000m, and at most two mountains are not higher than 4,000m.* This shows that *at most two mountains* is not an expression of type e. Explain why. (Your answer could take the form, "If this expression *were* of type e, we would expect ..., but instead we find the opposite: ...")

6.5 Quantifiers in object position

So far, we have restricted our attention to sentences with quantifiers in subject position, as in *Everybody likes Sam.* We have not considered sentences with quantifiers in object position, as in *Sam likes everybody.* As it happens, it is not possible to give an analysis of the latter type of sentences using Function Application alone. In Chapter 7, we will enrich our suite of compositional mechanisms so that we can deal with the so-called 'Problem of Quantifiers in Object Position' in a systematic way, with the use of a syntactic transformation rule called Quantifier Raising (QR) and a composition rule called Predicate Abstraction.

For now, let's assume that a transitive verb like *like* combines with a quantifier like *everybody* in the following manner:

(54)

$$\begin{array}{ccc} & \lambda x\,.\,\forall z[\mathsf{like}(x,z)] & \\ & \langle e,t\rangle & \\ \lambda y\lambda x\,.\,\mathsf{like}(x,y) & & \lambda P\,.\,\forall z P(z) \\ \langle e,\langle e,t\rangle\rangle & & \langle\langle e,t\rangle,t\rangle \\ \textit{like} & & \textit{everybody} \end{array}$$

In general, let us assume that any English expression like a transitive verb that translates as R of type $\langle e,\langle e,t\rangle\rangle$ can combine with a quantificational expression translated as Q of type $\langle\langle e,t\rangle,t\rangle$ to produce a type $\langle e,t\rangle$ predicate of the form $\lambda x\,.\,Q(\lambda z\,.\,R(z)(x))$. We can treat this as an ad-hoc composition rule called 'Quantificational Objects'.

Composition Rule 3. Quantificational Objects (QO)
Let γ be a syntax tree whose only two subtrees are α and β (in any order) where:

- $\alpha \rightsquigarrow R$ where R has type $\langle e,\langle e,t\rangle\rangle$
- $\beta \rightsquigarrow Q$ where Q has type $\langle\langle e,t\rangle,t\rangle$.

Then

$$\gamma \rightsquigarrow \lambda x\,.\,Q(\lambda z\,.\,R(z)(x))$$

This rule is a special case of a type-shifting rule we will discuss in Chapter 7.

Exercise 21. Using the rule of Quantificational Objects (QO), and the lexical entry for *nobody* given above, compositionally derive a meaning representation for *likes nobody*.

6.6 Negation and Quantifiers

6.6.1 Scope ambiguity

Consider the English sentence *Everyone doesn't like Sam.* What truth conditions does our current theory derive for this? If we use predicate negation marker presented in (30) 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 (55) below.

(55) $\forall x . \neg \mathsf{like}(x, \mathsf{s})$

That is indeed the most natural reading out of the blue. But context and prosody can bring out another reading:

(56) 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 the entire sentence, including the quantifier, as shown in (57) below.

(57) $\neg \forall x . \mathsf{like}(x, s)$

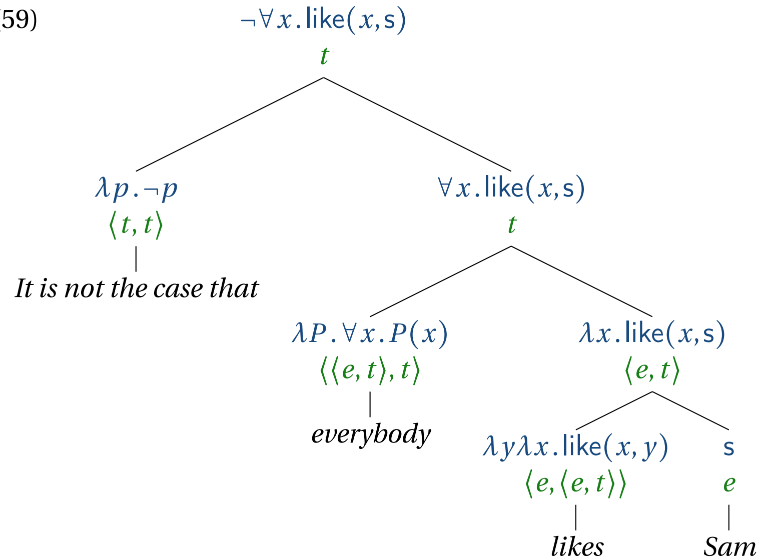
One way of accounting for this ambiguity is to positing two different meanings for negation. The first is **predicate negation**, which was presented in (30). Predicate negation negates the predicate in a subject-predicate structure, as in *Bjorn is not kind.* The second is **sentential negation**. In 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:

(58) *It is not true that* $\rightsquigarrow \lambda p . \neg p$

A derivation for *It is not true that everybody likes Sam* is shown in the following tree.

(59)



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:

(60) *not* $\rightsquigarrow \lambda p.\neg p$

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 (59), then we can derive the inverse scope reading for *Everybody doesn't like Sam.*

Sentences with quantifiers in object position are sometimes scopally ambiguous as well. Consider for example:

(61) 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 universal quantifier, and on the latter, the universal quantifier takes scope over negation.

Exercise 22. (a) Use predicate negation and the rule of Quantificational Objects to derive a representation for (61). 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 (61)?

6.6.2 Negative Concord

It is possible for a single sentence to contain both verbal negation and a negative indefinite such as *nobody*. In mainstream American English, such a sentence would have a **double negation (DN)** reading as shown in (62).

(62) Greta didn't see nobody.
$\neg\neg\exists x . \mathsf{see}(\mathsf{g}, x)$ (DN)

In a context where it was just asserted that Greta saw nobody, this sentence could be used to deny that claim, as in: *No, Greta didn't see nobody; Greta saw somebody.*

Assume that *see nobody* denotes a predicate that holds of z if there is no individual that z saw (using the rule of Quantificational

Objects):

(63) *saw nobody* $\rightsquigarrow \lambda z.\neg\exists x.\mathtt{see}(z, x)$

We then derive the formula $\neg\exists x.\mathtt{see}(\mathtt{g}, x)$ for *Greta saw nobody*. If *not* contributes negation on top of that, then we derive this double negation reading compositionally.

However, in many other languages (as well as some dialects of English) it is possible for a sentence to contain both verbal negation and a negative indefinite and still maintain a **single negation (SN)** reading. This phenomenon is called **negative concord**. Russian is an example of languages that has negative concord. In Russian, negation can be expressed using a negation marker *ne* that occurs preverbally:

(64) a. Greta spala.
Greta slept
'Greta slept'

b. Greta **ne** spala.
Greta not sleep
'Greta didn't sleep.'

An existential claim can be made using the quantifier *kogo-to* 'someone':

(65) Greta kogo-to uvidela.
Greta someone saw
'Greta saw someone'

The negation of this existential claim, the idea that that Greta did not see anybody, is not expressed simply by adding the preverbal negation marker to this sentence. Rather, a negative indefinite *nikogo* 'nobody' co-occurs with the preverbal negation marker, as in (66a). The negation marker is obligatory, as shown by the ungrammaticality of (66b).

(66) Russian

a. Greta ne videla nikogo.
Greta not saw nobody.
'Greta didn't see anybody' (SN)

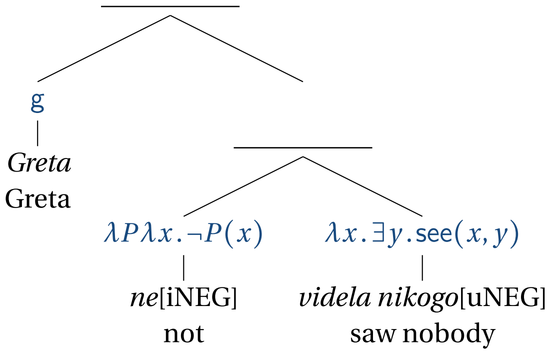
b. *Greta videla nikogo.
Greta saw nobody.

From the perspective we've built up so far, such cases present a puzzle: How is it that from the combination of two negative elements, only one negation is expressed?

Zeijlstra (2007) proposes that negative elements, such as verbal negation markers and negative indefinites, carry a negation feature that can be either interpretable ([iNEG]) or uninterpretable ([uNEG]), in the terminology of Minimalist syntax. Negative elements with interpretable negation features ([iNEG]) are semantically negative; they contribute negation as part of their lexical semantics. Negative elements with uninterpretable negation features ([uNEG]) are semantically non-negative, but their syntactic negation feature must be checked by an element carrying an interpretable negation feature higher up in the syntactic structure. A verbal negation marker like Russian *ne* is semantically negative if it expresses predicate or sentential negation; otherwise it just expresses an identity function. A quantifier like Russian *nikogo* is semantically negative if it is treated as synonymous with English *nobody*, and semantically non-negative if it is just an existential quantifier like English *somebody*. In the scope of negation, an existential quantifier yields the meaning of *nobody*.

In the case of Russian, Zeijlstra assumes that *ne* 'not' is semantically negative, bearing [iNEG], while *nikogo* 'nobody' is semantically non-negative (a mere existential quantifier), bearing [uNEG]. Zeijlstra assumes further that *nikogo* must be in the presence of a semantically negative item higher in the syntactic structure, because it carries an uninterpretable negation feature [uNEG] that must be checked. When the negative quantifier is in object position, *ne* (bearing [iNEG]) is able to fulfill this requirement. Hence we have the following structure:

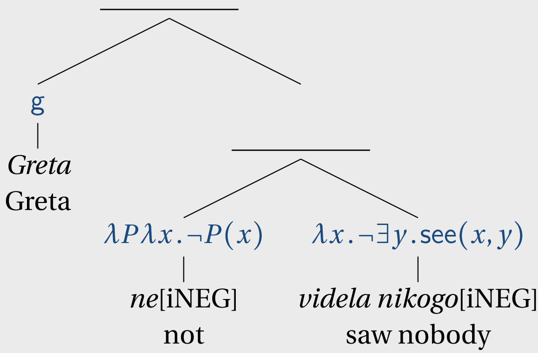
(67)



Exercise 23. Complete the derivation in (67). What sort of reading do you derive, single or double negation?

Exercise 24. Suppose that *nikogo* were semantically negative instead. What sort of reading would arise, single negation or double negation? Answer by means of filling in the following tree.

(68)



Now consider the Russian examples in (69), where *nobody* appears in subject position.

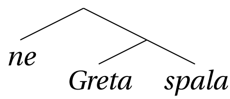
(69) Russian

a. Nikto ne spal.
Nobody not slept.
'Nobody slept' (SN)

b. *Nikto spal.
Nobody slept.

If *nikto* is [uNEG] as we have assumed for *nikogo*, then it requires an [iNEG] element to be present higher in the tree. On the surface, the verbal negation marker *ne* is lower in the tree than the negative indefinite in this case. But if we assume that negation is interpreted at a sentential level at LF, then at LF, the structure of (64b) is actually as in (70):

(70)

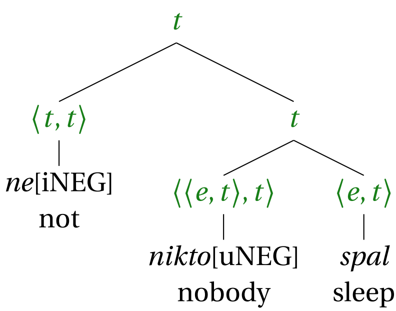


In that case, since *ne* is combining with a complete sentence instead of a verb phrase, it would make sense to analyze it as a function of type $\langle t, t\rangle$ that inverts the truth value of the input.

(71) *ne* $\rightsquigarrow \lambda p . \neg p$

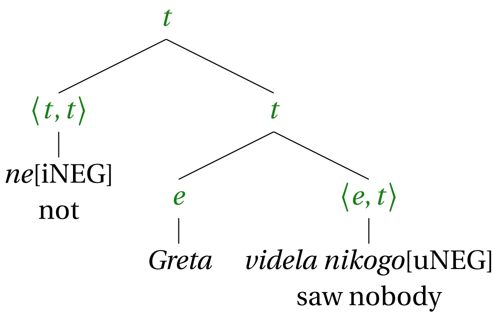
Hence the LF of *Nikto ne spal* 'Nobody slept' would be as follows:

(72)



The LF for *Greta ne videla nikogo* ‘Greta saw nobody’ would be:

(73)



Exercise 25. Complete the derivations in (72) and (73). For both sentences, provide an English paraphrase of the reading that you compositionally derive.

Exercise 26. Give a lexical entry for *nikogo* ‘nobody’ with type $\langle\langle e, t\rangle, t\rangle$, and assume that *ne* expresses **predicate negation**, and is interpreted in its surface position in the tree. Using your lexical entry, give a translation for the VP *videla nikogo* (as in *Greta ne videla nikogo*). Then give a compositional analysis of (64a), (64b), (65), and (66a). For both sentences, provide an English paraphrase of the reading that you compositionally derive.

6.7 Generalized quantifiers

(This section is under development.)

We have said that *every cat* translates to the following expression of type $\langle\langle e, t\rangle, t\rangle$:

(74) *every cat* $\rightsquigarrow \lambda P\,.\,\forall x\,.\,\mathsf{Cat}(x) \rightarrow P(x)$

What does this expression denote? There are several equivalent ways to think about this question. One way is as a function from predicates to truth values. Taking P to be a variable over predicates, (74) denotes the function that maps those predicates that apply to every cat to True, and all other predicates to False. This denotation corresponds to a set of sets of entities: (74) denotes the set of all sets that contain every cat—that is to say, the set of all supersets of the set of cats. Writing CAT for the set of cats, and (as usual) D_e for the domain of entities, we can write this set as follows:

(75) $\{P \subseteq D_e : \mathsf{CAT} \subseteq P\}$

Similarly, *some dog* translates to this:

(76) *some dog* $\rightsquigarrow \lambda P\,.\,\exists x\,.\,[\mathsf{Dog}(x) \wedge P(x)]$

which denotes the set of all sets that are not dog-free:

(77) $\{P \subseteq D_e : P \cap \mathsf{DOG} \neq \varnothing\}$

We will call these sets $\mathsf{everyCat}$ and $\mathsf{someDog}$. They can be visualized as in Figures 6.2 and 6.3.

Let us assume that the noun *thing* denotes D_e in every model, that is, it always denotes the universal property (the predicate that applies to all entities). (We ignore the fact that in practice it is a bit odd to refer to people and other animate beings as things.) By replacing CAT and DOG with D_e, we arrive at plausible denotations for the English words *everything* and *something* and the English

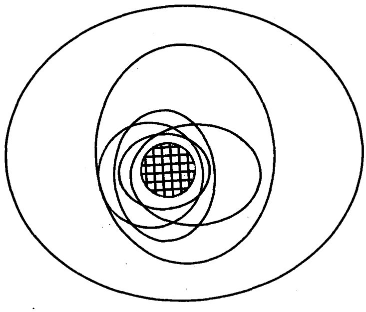


Figure 6.2: The generalized quantifier everyCat denoted by *every cat*. The biggest circle represents the universe of discourse. The cross-hatched circle represents the set of cats. The other circles represent some of the sets in the denotation of the generalized quantifier. The cross-hatched circle must be fully contained in each of the other circles because they represent properties common to *every* cat, that is, supersets of the set of cats.

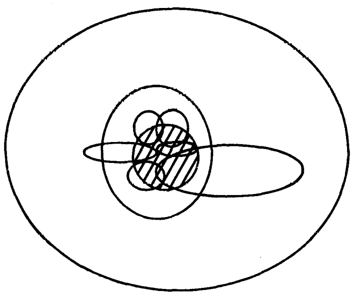


Figure 6.3: The generalized quantifier someDog denoted by *some dog*. The biggest circle represents the universe of discourse. The dashed circle represents the set of dogs. The other circles represent some of the sets in the denotation of the generalized quantifier. The cross-hatched circle need not be fully contained within any of the other circles, because they represent properties of *some* dogs.

expressions *every thing* and *some thing*. We will call these everyThing and someThing:

(78) a. everyThing $=_{def} \{P \subseteq D_e : D_e \subseteq P\} = \{D_e\}$
b. someThing $=_{def} \{P \subseteq D_e : P \cap D_e \neq \varnothing\}$
$= \{P \subseteq D_e : P \neq \varnothing\}$

These sets are, respectively, the singleton set of D_e, and the set of nonempty subsets of D_e. We can understand them as conditions on properties. In order for a property P to be included in everyThing, it has to be the universal property. In order for P to be included in someThing, it merely has to be nonempty. In the same vein, we can represent the generalized quantifiers noThing, which is denoted by the expressions *nothing* and *no thing*; exactlyTwoThings, which is denoted by *exactly two things*; and atLeastTwoThings, which is denoted by *at least two things*, *more than one thing*, and *two or more things*.

(79) a. noThing $=_{def} \{P \subseteq D_e : |P| = 0\} = \{\varnothing\}$
b. exactlyTwoThings $=_{def} \{P \subseteq D_e : |P| = 2\}$
c. atLeastTwoThings $=_{def} \{P \subseteq D_e : |P| \geq 2\}$

This says that for a set P to be included in noThing, it has to be the empty set; for exactlyTwoThings, it has to contain exactly two things; and for atLeastTwoThings, it has to contain at least two things.

Which quantifier does *two things* denote? It is clear that *Two things are blue* implicates *Exactly two things are blue*, but what is the status of this implication? Most semanticists take it to be an implicature, as opposed to an entailment. If this is correct, *two things* denotes atLeastTwoThings in (79c). Otherwise, it denotes exactlyTwoThings in (79b).

Sets of sets of entities like those in (79) are called GENERALIZED QUANTIFIERS. This is because they generalize the standard quantifiers $\forall$ and $\exists$ of first-order logic.

Some generalized quantifiers are FIRST-ORDER DEFINABLE; that is, they are the denotations of lambda expressions whose value descriptions are built up using the rules of first-order logic only. This includes the denotations of *every cat* in (74) and *some dog* in (76), as well as the quantifiers in (79). Some of these look very simple:

(80) *nothing* $\rightsquigarrow \lambda P. \neg \exists x. P(x)$

Others look quite unwieldy:

(81) *exactly two things* $\rightsquigarrow \lambda P. \exists x. \exists y. \neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg \exists z. P(z) \wedge \neg(z = x) \wedge \neg(z = y)$

Other generalized quantifiers, such as those denoted by *most swans, most things, one in three cats* are not 'first-order definable' in the sense that they cannot be expressed in first-order logic. But they can be defined in terms of sets:

(82) a. $\mathsf{mostSwans} =_{def} \{P \subseteq D_e : |\mathsf{SWAN} \cap P| > |\mathsf{SWAN} - P|\}$
b. $\mathsf{mostThings} =_{def} \{P \subseteq D_e : |P| > |D_e - P|\}$
c. $\mathsf{oneInThreeCats} =_{def} \{P \subseteq D_e : |P \cap \mathsf{CAT}|/|\mathsf{CAT}| = 1/3\}$

The fact that these generalized quantifiers are not first-order definable doesn't prevent us from defining them in L_λ, since this is a language of higher-order logic. One way to do this would be to include numbers into our domains as a new basic type in addition to entities and truth values, as well as functions from sets of entities to numbers (like cardinality), operations on numbers (like /, the division operation), relations between numbers (like >, the greater-than relation), and meaning postulates that ensure that all these things behave in the ordinary mathematical sense. But with all these additions, our logical language L_λ would become cumbersome. Instead, to keep L_λ simple, we will just regard the formal definitions in (79) and similar ones below as part of our meta-language.

The denotations of noun phrases like *every cat* and *most swans* are built up compositionally, at least from those of the words they consist of. (It has even been suggested that *most* is internally complex and can be seen as a combination of the comparative *more* and the superlative morpheme *-st*, which is the same morpheme that we find on the end of words like *tallest* and *funniest*.) So determiners denote functions from nouns (type $\langle e,t\rangle$) to generalized quantifiers (type $\langle\langle e,t\rangle,t\rangle$), or in other words, functions of type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$. We will refer to functions of this type as DETERMINER FUNCTIONS or just DETS.

The following translation of *every* denotes a Det that yields the generalized quantifier everyCat when applied to the denotation of *cat*:

(83) *every* $\rightsquigarrow \lambda P'\lambda P\,.\,\forall x\,.\,P'(x) \rightarrow P(x)$

The Det that this denotes is the left-to-right Curried version of the subset relation. We will refer to this relation as every:

(84) every $=_{def} \{\langle P',P\rangle \mid P' \subseteq P\}$

Likewise, the translation of *some* in (85) denotes the Curried version of the relation in (86), which holds between any two sets when they overlap:

(85) *some* $\rightsquigarrow \lambda P'\lambda P\,.\,\exists x\,.\,P'(x) \wedge P(x)$

(86) some $=_{def} \{\langle P',P\rangle \mid P' \cap P \neq \varnothing\}$

This function maps the denotation of *dog* to the generalized quantifier someDog. We will refer to it as some. In general, we will call each Det we discuss by the determiner that denotes it, or that comes closest to denoting it.

The first argument of a Det is called its RESTRICTOR, and its second argument, its NUCLEAR SCOPE. The noun that a determiner combines with is called the restrictor of that determiner because, intuitively, it restricts the attention of the noun phrase to just those entities to which the noun applies.

In a sentence like *Every cat meows*, the word *every* denotes the Det every, whose restrictor is denoted by *cat*, and whose nuclear scope is denoted by *meows*.[3]

Unlike $\exists$ and $\forall$, some and every do not bind any variables. But some and every give rise to formulas with the same truth conditions as the first-order logic quantifiers $\exists$ and $\forall$:

(87) a. $\mathsf{some}(\lambda x.\mathsf{Dog}(x))(\lambda x.\mathsf{Barks}(x))$
b. $\exists x.\mathsf{Dog}(x) \wedge \mathsf{Barks}(x)$

(88) a. $\mathsf{every}(\lambda x.\mathsf{Cat}(x))(\lambda x.\mathsf{Meows}(x))$
b. $\forall x.\mathsf{Cat}(x) \rightarrow \mathsf{Meows}(x)$

So in first-order logic, quantification and variable binding are conflated, but in the case of generalized quantifiers they come apart.

The Dets some and every are special not just because they replicate $\exists$ and $\forall$, but also because they are SORTALLY REDUCIBLE. This means that they each denote a relationship between two sets that can be expressed using just the set theoretic operations of intersection, union, and complement. These operations correspond to the propositional logic connectives. This is why we were able to translate some using $\wedge$ and intersection, and every using $\rightarrow$ and subsethood.

We can test whether a Det is sortally reducible by looking for paraphrases of determiners where the restrictor is replaced by *entity* and the nuclear scope incorporates the old restrictor and the expressions *and, or, not, if . . . then* (read as the material conditional) and *if and only if*:

(89) a. Some A is a B $\Leftrightarrow$
b. Some entity is both an A and a B.

(90) a. Every A is a B $\Leftrightarrow$

[3] In the literature, generalized quantifiers are also called Type (1) quantifiers because they combine with one unary function (their nuclear scope), and Dets are also called Type (1,1) quantifiers because they combine with two unary functions (their restrictor and their nuclear scope).

b. Every entity is such that, if it is an A, then it is a B.

A Det that is not sortally reducible is called INHERENTLY SORTAL. An example is most, which cannot be paraphrased in the required way:

(91) a. Most As are Bs. $\not\Leftrightarrow$
b. Most entities are such that, if they are As, then they are Bs.

To illustrate, take A to be the set of swans, and B to be the set of black entities. In a model where there are 8 white swans, 2 black swans, and 90 ducks, sentence (91a) is intuitively judged false. But sentence (91b), with *if* read as the material conditional, is true, because each duck makes that the material conditional vacuously true.

The Dets some and every are examples of Dets called INTERSECTIVE and CO-INTERSECTIVE. These two classes of Dets taken together form the class of sortally reducible Dets. The four sets that a Det can depend on are represented visually in the Venn diagram in Figure 6.4 and labeled $A \cap B$, $A - B$, $B - A$, and $(A \cup B)'$, where A is the restrictor and B is the nuclear scope. The union of these four sets is called the UNIVERSE OF DISCOURSE. This is the same as the domain of individuals D_e.

An INTERSECTIVE Det depends only on $A \cap B$, the intersection of its restrictor and nuclear scope. For example, some is intersective because in order to know whether *Some As are Bs* is true, all you need to know is something about the set $A \cap B$—in this case, whether it is nonempty. By contrast, every is not intersective, because even if you know precisely which entities are in $A \cap B$, you don't yet know whether *Every A is a B* is true. For that, you would need to know whether there are any entities in $A - B$, the set of entities that are As without being Bs. Is this all you need to know? This depends on whether *Every A is a B* has EXISTENTIAL IMPORT, that is, whether it entails *Some A is a B*. If so, then you need to

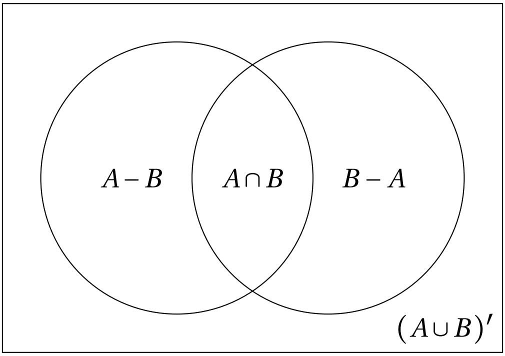


Figure 6.4: The four sets that a Det can depend on.

know not only whether $A - B$ is empty, but also whether $A \cap B$ is. In his syllogistic logic, Aristotle treated universal quantifiers as having existential import; in first-order logic, they don't; and as for whether this holds in natural language, we come back to it in Chapter 8. As defined in (84), every is intended to mirror the behavior of $\forall$ in first-order logic, and therefore lacks existential import.

Exercise 27. Define a version of every that has existential import and call it every$_\exists$.

A CO-INTERSECTIVE Det depends on $A - B$ and nothing else. So every as defined in (84) is CO-INTERSECTIVE, while every$_\exists$ as defined in Exercise **??** would be neither intersective nor co-intersective. As another example, most is neither intersective nor co-intersective, since in order to know whether most As are Bs one needs to know something about $A \cap B$ and about $A - B$ (namely, whether the first set has more members than the second).

While some and every differ in which set they depend on, they have something in common: they depend only on the cardinality of that set, and not on the identity of its members. A CARDINAL

Det depends just on the cardinality of $A \cap B$, and a CO-CARDINAL one depends just on the cardinality of $A - B$. Some depends on whether the cardinality of $A \cap B$ is nonzero, and every depends on whether the cardinality of $A - B$ is zero. Cardinal Dets include some, a, no, practically no, more than ten, fewer than ten, exactly ten, about ten, ten or more, between ten and twenty, and so on. Co-cardinal Dets include every and all but two.

The Det most is neither cardinal nor co-cardinal; it is PROPORTIONAL. A proportional Det depends on the proportion of the cardinalities of the sets $A \cap B$ and $A - B$, and on nothing else; for example, most depends on whether that proportion is greater than half. Other proportional Dets are at least half the, ten percent of the, less than two-thirds of the, etc. Proportional Dets are not first-order definable.

So far we have seen examples of Dets that depend only on $A \cap B$, Dets that depend only on $A - B$, and Dets that depend on both. As Fig. 6.4 shows, there are two more sets that Dets could in principle depend on: $B - A$, and $(A \cup B)'$. Dets which do not depend on $B - A$ are called CONSERVATIVE, and Dets that do not depend on $(A \cup B)'$ satisfy EXTENSION. One can prove that the Dets that satisfy conservativity and extension are just those which relativize a generalized quantifier. (To RELATIVIZE a generalized quantifier means to convert it into a determiner which behaves like the generalized quantifier in question after it combines with its first argument. For example, the determiner every relativizes the generalized quantifier everything, assuming that *thing* ranges over the entire universe of discourse.) Most semanticists agree that all Dets denoted by determiners, as well as comparable lexical items in any natural language, satisfy both conservativity and extension. This has been proposed as a SEMANTIC UNIVERSAL, a property that holds across all languages (Barwise & Cooper, 1981).

All determiners we have discussed so far conform to this universal. What would a Det look like that violates it? One example is the Det in (92), which violates conservativity but satisfies extension:

(92) $\{\langle A,B\rangle \mid |A| = |B|\}$

The English word that is perhaps closest in meaning to this Det is the adjective *equinumerous* (i.e. of equal number). If it could be used as a determiner in a sentence like *Equinumerous cats are dogs*, to express that there are as many cats as there are dogs, it would be a counterexample to the conservativity universal. But this sentence is not grammatical.

As another example, a Det that means the same as *some* on universes of discourse with fewer than five elements, and the same as *at least five* otherwise, would obey conservativity but would violate extension.

The following schema can be used to test whether a determiner denotes a conservative Det:

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

For example, *most* denotes a conservative determiner because the following is a valid statement:

(94) Most dogs bark iff most dogs are dogs that bark.

To better understand what it would mean for a Det not to be conservative, consider the word *Only*:

(95) Only dogs bark iff only dogs are dogs that bark.

This is not a valid statement. The left-hand side may well be false, but the right-hand side will always be true. For example, in a model in which dogs and sea lions bark, it is not true that only dogs bark; but it is always true that only dogs are dogs that bark.

This suggests that the word *only* does not denote a conservative Det. If anything, it denotes a Det like the following:

(96) $\{\langle A,B\rangle \mid B \subseteq A\}$

In the case of this Det, knowing just $A \cap B$ and $A - B$ is not enough. Rather, one would need to know whether $B - A$ is empty. And this is precisely what the truth of *Only As are Bs* hinges on.

While the word *only* bears some resemblance to a determiner, most linguists do not regard it as such, because it has a wider distribution than determiners. For example, it composes not only with nouns as determiners do, but also with verb phrases and noun phrases:

(97) a. John only read two papers today.
b. Only the postman rang twice.

For similar reasons, other putative counterexamples to the conservativity universal such as *just* and *mostly* are generally not seen as genuine.

Formulating crosslinguistic generalizations such as the conservativity universal is only one example of the many applications of generalized quantifier theory in linguistics. Another one is the distribution of negative polarity items, which we discussed in Chapter 2. Other applications account for the restricted distribution of noun phrases in various constructions. An example is the EXISTENTIAL-THERE CONSTRUCTION, a construction which is used to talk about existence or nonexistence and which consists of the word *there*, an inflected form of *be*, a noun phrase called the PIVOT, and typically a CODA such as a prepositional phrase. Here are some examples:

(98) a. There were four men at the table.
b. There is a unicorn in the garden.
c. There was nobody in the building.
d. There are a lot of books regarding this.
e. There were three or more voting members present.
f. There are the same number of students as teachers on the committee.

The question is which noun phrases can and cannot be used as pivots. The following examples are infelicitous.

(99) a. #There was John at the table.
b. #There are most angels in heaven.
c. #There was everybody in the building.
d. #There are both books regarding this.
e. #There were the three or more voting members present.
f. #There are two out of three students on the committee.

(We set aside sentences like *There is the problem of the cockroaches escaping*, which present instances of something whose existence has already been asserted or implied.)

Generalized quantifier theory provides an elegant account of this problem. As a first approximation, the noun phrases that occur as pivots in existential-there sentences are just those that are intersective.

Another linguistic application concerns PARTITIVES, that is, noun phrases with two determiners separated by the word *of*. The question is which determiners can occur on the left and on the right of *of*:

(100) a. two of the students
b. most of these cats
c. half of John's books
d. some of your cookies
e. each of the five examples

(101) a. ?two of most students
b. *two of each student
c. *none of no tables
d. *half of ten people
e. *the of the five examples
f. *the two of the five examples

The relevant notion is DEFINITENESS. Noun phrases headed by determiners such as *the*, such as *the woman* or *the moon*, are DEF-

INITE, as opposed to INDEFINITE noun phrases like *a star, some man,* or *three women.* Definite determiners are excluded from the left of *of*, but not from the right; in fact, it is sometimes claimed that it is only definite determiners that can appear on the right of *of.* Definiteness is usually described in terms of familiarity and uniqueness. FAMILIARITY means that the referents of definite noun phrases have been previously introduced in the discourse (e.g. *the woman* refers to a woman previously mentioned or otherwise made salient), while UNIQUENESS means that there is only one item matching the description (e.g. *the moon* works because there is only one moon). The notion of uniqueness doesn't work for plural definites such as *the stars* or *the three little pigs.* But it is possible to define a definite Det in a way that extends to these cases. We come back to this question in Chapter 8.

6.8 Toy fragment

So far, we have developed the following toy fragment of English, consisting of a set of syntax rules, a lexicon, a set of composition rules, and a set of lexical entries.

Syntax

S	→	DP VP
S	→	S CoordP
CoordP	→	Coord S
VP	→	V (DP|AP|PP|NegP)
NegP	→	Neg (VP|AP)
AP	→	A (PP)
DP	→	D (NP)
NP	→	N (PP)
NP	→	A NP
PP	→	P DP

Lexicon

Coord: *and, or*
Neg: *not*
V: *smiled, laughed, loves, hugged, is*
A: *Swedish, happy, kind, proud*
N: *singer, drummer, musician*
D: *the, a, every, some, no*
D: *Agnetha, Frida, Björn, Benny, everybody, somebody, nobody*
P: *of, with*

Composition rules

- **Function Application** (FA)
  Let γ be a tree whose only two subtrees are α and β where:

- $\alpha \leadsto \alpha'$ where α' has type $\langle\sigma,\tau\rangle$
- $\beta \leadsto \beta'$ where β' has type σ.

Then

$$\gamma \leadsto \alpha'(\beta')$$

- **Non-branching Nodes** (NN)
If β is a tree whose only daughter is α, where $\alpha \leadsto \alpha'$, then $\beta \leadsto \alpha'$.

- **Quantificational Objects** (QO)
Let γ be a syntax tree whose only two subtrees are α and β (in any order) where:

  - $\alpha \leadsto R$ where R has type $\langle e,\langle e,t\rangle\rangle$
  - $\beta \leadsto Q$ where Q has type $\langle\langle e,t\rangle,t\rangle$.

  Then

$$\gamma \leadsto \lambda x.Q(\lambda z.R(z)(x))$$

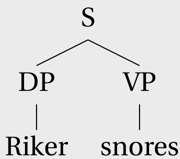
Lexical entries

- *Agnetha* $\leadsto \mathsf{a}$
- *smiled* $\leadsto \lambda x.\mathsf{smiled}(x)$
- *loves* $\leadsto \lambda y\lambda x.\mathsf{loves}(x,y)$
- *kind* $\leadsto \lambda x.\mathsf{kind}(x)$
- *is* $\leadsto \lambda P.P$
- *with* $\leadsto \lambda y\lambda x.\mathsf{with}(x,y)$
- *of* $\leadsto \lambda x.x$
- *a* $\leadsto \lambda P.P$
- *not* $\leadsto \lambda P\lambda x.\neg P(x)$

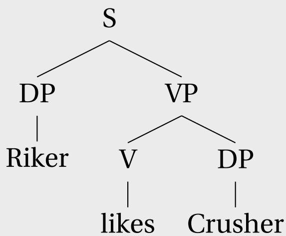
- *some* $\rightsquigarrow \lambda P \lambda P' . \exists x . [P(x) \wedge P'(x)]$
- *no* $\rightsquigarrow \lambda P \lambda P' . \neg \exists x . [P(x) \wedge P'(x)]$
- *every* $\rightsquigarrow \lambda P \lambda P' . \forall x . [P(x) \rightarrow P'(x)]$
- *something* $\rightsquigarrow \lambda P . \exists x . P(x)$
- *nothing* $\rightsquigarrow \lambda P . \neg \exists x . P(x)$
- *everything* $\rightsquigarrow \lambda P . \forall x . P(x)$

Exercise 28. Give fully beta-reduced translations at each node of the following trees. Provide appropriate lexical entries as needed.

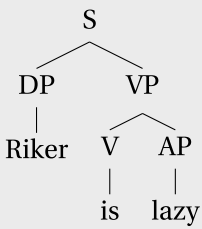
(a)



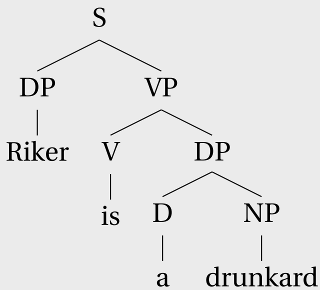
(b)



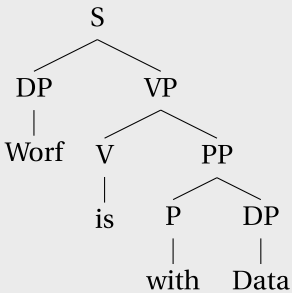
(c)



(d)

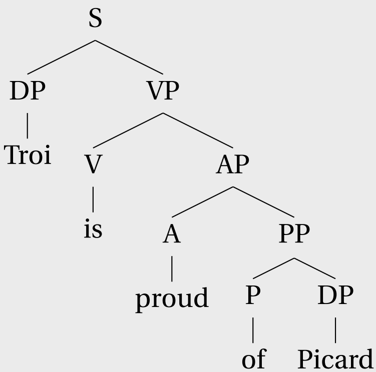


(e)

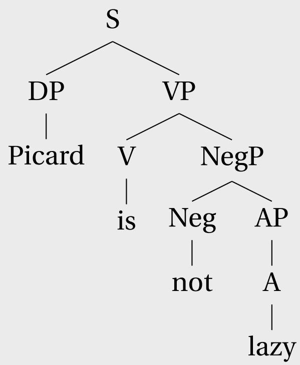


Exercise 29. Give fully beta-reduced translations at each node of the following trees. Provide appropriate lexical entries as needed.

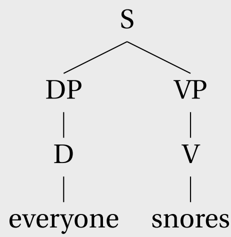
(a)



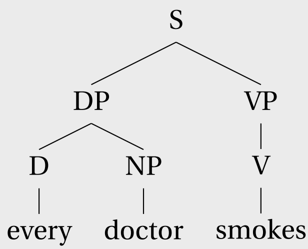
(b)


S
DP
VP
Picard
V
NegP
is
Neg
AP
not
A
lazy

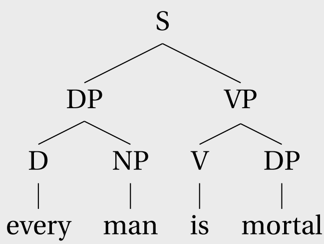
(c)


S
DP
VP
D
V
everyone
snores

(d)


S
DP
VP
D
NP
V
every
doctor
smokes

(e)


S
DP
VP
D
NP
V
DP
every
man
is
mortal

Exercise 30. Extend the fragment to assign representations in L_λ to the following sentences. For both sentences, give a parse tree with a fully beta-reduced representation at each node.

- *Björn is a fan of Agnetha.*
- *Björn is Agnetha's fan.*

The two representations should be equivalent to each other, in order to capture the fact that the English sentences are.

Exercise 31. Extend the fragment to assign representations in L_λ to the following sentences so that the following two sentences are equivalent. For both sentences, give a parse tree with a fully beta-reduced representation at each node.

- *Benny smokes and Benny drinks.*
- *Benny smokes and drinks.*

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 two new composition rules: Predicate Modification and Predicate Abstraction.

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)
$\therefore$ This doubt is reasonable. (Conclusion, predicative use)

(4) This doubt is reasonable. (Premise, predicative use)
$\therefore$ 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)
$\therefore$ This is a doubt. (Conclusion)

Lastly, we can reason from the adjective and the noun to their combination (at least on the judge's interpretation of *reasonable*):

(6) This is reasonable. (Premise 1)
This is a doubt. (Premise 2)
$\therefore$ This is a reasonable doubt. (Conclusion)

We will abbreviate these three entailment patterns with the following statement:

(7) This is a reasonable doubt iff this is reasonable and this is a doubt.

There are many other adjectives that give rise to the same entailment pattern:

(8) Frida is a Norwegian millionaire iff Frida is Norwegian and Frida is a millionaire.

(9) John is a vegetarian farmer iff John is vegetarian and John is a farmer.

How do we explain entailment patterns like these? A simple answer is that the adjectives (*reasonable*, *Norwegian*, and *vegetarian*) denote sets—the set of all reasonable things, for example—

and that the nouns (*doubt*, *millionaire*, and *farmer*), just like other nouns we have seen, denote sets as well—for example, the set of doubts. A reasonable doubt, then, is something which is in each of these two sets, or in other words, in their intersection. Adjectives that combine with nouns in this way are also called INTERSECTIVE ADJECTIVES.

One of the hallmarks of intersective adjectives is that they make arguments of the following form valid:

(10) John is a vegetarian farmer.
John is a doctor.
∴ John is a vegetarian doctor.

This is as predicted based on what we have said about the denotation of intersective adjectives: If being a vegetarian farmer is nothing more and nothing less than being both vegetarian and a farmer, and being a vegetarian doctor is just being a vegetarian and being a doctor, then any vegetarian farmer who is also a doctor should count as a vegetarian doctor.

Many adjectives are not intersective. For example, Albert Einstein was not only an outstanding physicist but also an amateur violinist. The following argument is grammatically parallel to the one in (10) but not valid:

(11) Einstein is an outstanding physicist.
Einstein is a violinist.
$\not\therefore$ Einstein is an outstanding violinist.

However, just as with *reasonable*, dropping the adjective results in a valid argument:

(12) Einstein is an outstanding physicist.
∴ Einstein is a physicist.

The validity of (12) leads us to conclude that *outstanding physicist* denotes a subset of what *physicist* denotes, just as *amateur vio-*

linist denotes a subset of the denotation of *violinist*. In this sense, both *outstanding* and *amateur* are SUBSECTIVE ADJECTIVES. But *outstanding* is not an INTERSECTIVE ADJECTIVE. If it were, then an outstanding physicist who is also a violinist should also be an outstanding violinist.

Some phrases appear to be ambiguous between an intersective and a non-intersective reading. A famous example is *beautiful dancer*:

(13) 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:

(14) John is an alleged murderer.
$\not\therefore$ 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.

Adjectives like *counterfeit*, *fake*, and maybe *former* are special among non-subsective adjectives in that they seemingly map sets to disjoint sets (they are also called PRIVATIVE). For example, while some alleged murderers really are murderers, no fake gun really is a gun. Or is it? This depends on which set the noun *gun* is taken to denote: either the set of real guns, or the set of real and fake guns taken together. If the following entailment is valid, that would suggest that only real guns are included:

(15) This is a fake gun.
$\therefore$ This is not a gun.

On the other hand, if only real guns are included, then it is not clear why sentences like the following have nontrivial meanings:

(16) This gun is fake.

(17) Is this gun real or fake?

We will not settle this issue here.

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.

7.2 Adjectives

The logical counterpart of intersection is conjunction. From the conjunction $[A \wedge B]$ we can reason to A and to B and back, so using conjunction to translate sentences with attributively-used intersective adjectives will explain their entailment patterns. (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 in Chapter 12.)

(18) This is a reasonable doubt.
$\mathsf{reasonable}(\mathsf{this}) \wedge \mathsf{doubt}(\mathsf{this})$

(19) This is reasonable.

$\mathsf{reasonable(this)}$

(20) This is a doubt.
$\mathsf{doubt(this)}$

We will treat *outstanding* and other subsective adjectives as functions from sets to subsets. To do so, we rely on a higher-order function term $\mathsf{outstandingAs}$ of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. For any type $\langle e,t\rangle$ expression P, $\mathsf{outstandingAs}(P)$ is a new expression of type $\langle e,t\rangle$.

In order to ensure that $\mathsf{outstandingAs}$ denotes a function from sets to subsets, and thus that 'x is an outstanding P' entails 'x is a P', we can stipulate that the following formula must be true in every model:

(21) $\forall P \forall x . \mathsf{outstandingAs}(P)(x) \rightarrow P(x)$
(For every set P, every outstanding P-individual is a P-individual.)

What we have written in (21) is not part of any lexical entry; it is a constraint that every model must satisfy. This kind of assumption is what Montague called a MEANING POSTULATE. Another example of a meaning postulate would be a constraint requiring that every $\mathsf{bachelor}$ is Male, no matter the circumstances. This kind of constraint is a way of capturing the fact that being male is part of what it means to be a bachelor (hence the term 'meaning postulate'). What is encoded in (21) is that being P is part of what it means to be 'outstanding as a P'.

Non-subsective adjectives like *alleged* can be treated in the same way as subsective adjectives except without a meaning postulate like this. Without this meaning postulate, no entailment from sentences of the form 'x is an [adjective] [noun]' to 'x is a [noun]' is predicted.

With this in place, let us assume the following translations (where $\rightsquigarrow$ signifies the translation in question):

(22) Einstein is an outstanding physicist.
$\rightsquigarrow$ outstandingAs(physicist)(e)

(23) Einstein is a physicist.
physicist(e)

Using these assumptions, we can explain why *Einstein is an outstanding physicist* implies *Einstein is a physicist*: The formula in (23) follows logically from the formula (22), together with the meaning postulate in (21). (The corresponding entailment for non-subsective adjectives is blocked because the meaning postulate is absent in those cases.)

Now, suppose we take *Einstein is outstanding* to mean that Einstein is outstanding in some salient respect; then its translation would be as follows:

(24) Einstein is outstanding.
outstandingAs(P)(e)

This translation contains a free variable P, whose interpretation needs to be specified by the contextually supplied assignment function. (This is just one of several potentially viable ways to treat adjectives like *outstanding*.) This captures the fact that being outstanding as a physicist does not entail being outstanding unconditionally, at least not in every context.

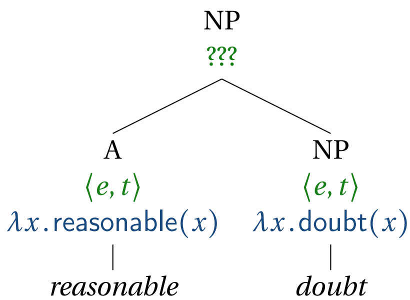
Exercise 1. Explain how the treatment of *outstanding* given above blocks the inference from *Einstein is an outstanding physicist* to *Einstein is an outstanding violinist.*

Having translated our sentences into logic, we have accounted for the entailment relations between them. But how do we make this translation compositional? Let us first consider intersective adjectives, since these are the simplest case, and then see what needs to change so we can account for other types of adjectives.

In the previous chapter, we considered sentences with adjectives and nouns like *Björn is kind* and *Agnetha is a singer*, treating *kind* and *singer* as type $\langle e,t\rangle$. So we know how to derive truth conditions compositionally for (19) and (20).

But we do not yet have the tools to analyze sentences like *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.

(25)

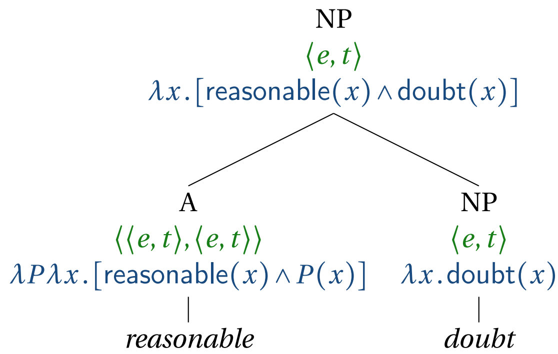


At this point, one might try and adopt a different denotation for *reasonable*, one that can be applied directly to *doubt*. This denotation would expect a predicate like *doubt*, and return a new predicate that holds of individuals that are both reasonable and in the set denoted by the input predicate. Such an expression would be of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. That is, its input type and its output type are the same.

(26) *reasonable* $\rightsquigarrow \lambda P.\lambda x.[\text{reasonable}(x) \wedge P(x)]$

This expression avoids the type mismatch in (25):

(27)



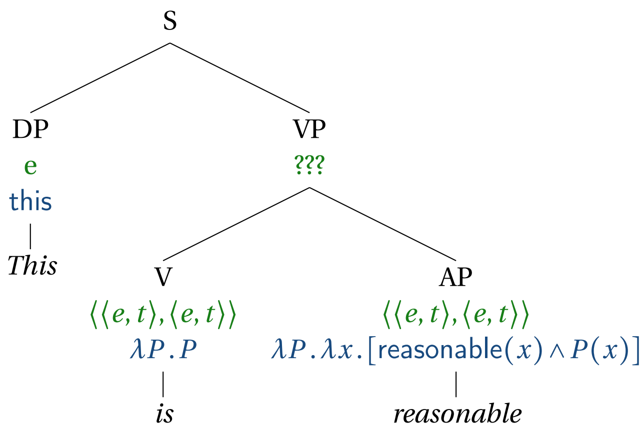
We will call this second translation for *reasonable* a MODIFIER and its type a MODIFIER TYPE. In Chapter 11, we will encounter another category that can be analyzed as having modifier types: adverbs.

The drawback of this translation is that it does not work smoothly for intersective adjectives in predicative position:

(28) This is reasonable.

Here the modifier-type translation above leads to a type mismatch:

(29)



So the modifier type analysis causes problems for adjectives in predicative position. We adopted it to solve the type mismatch in attributive positions, where the adjective applies to a noun. But in predicative position, there is no noun for the adjective to apply to. The type of the identity function denoted by *is* is the same as that of *reasonable*, so the two don't combine. Even if we ignored *is* or allowed it to apply to functions of arbitrary types, the resulting VP denotation would still not be of the right type to combine with the subject, and it would expect one too many arguments. So at this point we have considered two translations, and each one works fine for one position but does not work for the other.

There are at least two ways to resolve this problem. We can (i) generate two translations for the adjective *reasonable* (and similarly for other intersective adjectives): one of type $\langle e,t\rangle$ for predicative positions, and another one of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ for attributive positions. Or (ii), we can give intersective adjectives a single translation no matter which position they occur in, and eliminate the type mismatches by introducing a new composition rule. While the two ways lead to the same result, each one involves tools that have many other uses beyond adjectives, so we will consider them both.

To implement option (i) and capture the semantic relationship between attributive and predicative uses of adjectives, we take one translation to be basic and derive the other one from it with the help of either a TYPE-SHIFTING RULE or a SILENT OPERATOR. Type-shifting rules and silent operators are theoretical devices that generate additional translations/denotations for a given constituent. The difference between them is that type-shifting rules are typically regarded as invisible to the syntactic component of the grammar; by contrast, silent operators are generally assumed to have a reflection in the syntax.

For concreteness, we will take the translations of type $\langle e,t\rangle$ to be basic and those of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ to be derived. The basic type $\langle e,t\rangle$ is the right one for predicative positions, as we have seen in the previous chapter for analogous sentences to *This doubt is reasonable*. For attributive positions (*reasonable doubt*), we will now derive translations of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. If we use a silent operator MOD to generate derived translations, we will represent this as follows:

(30)
A
$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
$\lambda P.\lambda x.[\text{reasonable}(x) \wedge P(x)]$

- $\langle\langle e,t\rangle,\langle\langle e,t\rangle,\langle e,t\rangle\rangle\rangle$
  $\lambda P'\lambda P.\lambda x.P'(x) \wedge P(x)$
  - MOD
- A
  $\langle e,t\rangle$
  $\lambda x.\text{reasonable}(x)$
  - *reasonable*

where MOD is an unpronounced word. Alternatively, we can use the following type-shifting rule to equivalent effect:

Type-Shifting Rule 1. Predicate-to-modifier shift (MOD)
If $\alpha \rightsquigarrow \alpha'$, where α' is of type $\langle e,t\rangle$,
then $\alpha \rightsquigarrow \lambda P.[\alpha'(x) \wedge P(x)]$ (as long as P and x are not free in α'; in that case, use different variables of the same type).

We will represent the application of this rule in the syntax tree like this:

(31)
A
$\langle\langle e,t\rangle,\langle e,t\rangle\rangle$
$\lambda P.\lambda x.\text{reasonable}(x) \wedge P(x)$
$\Uparrow$MOD
A
$\langle e,t\rangle$
$\lambda x.\text{reasonable}(x)$
|
reasonable

Our notation uses the upwards-facing double arrow $\Uparrow$ in order to capture the intuition that the type-shifting operation induces a transformation of the denotation.

Exercise 2. We could go the other way around in principle, and take $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ as the basic type and $\langle e,t\rangle$ as the derived type. This requires introducing either a silent operator with a trivial translation such as $\lambda x.x = x$, which denotes the set of all individuals, or a type shifting rule with the same effect. Specify what the type shifting rule would look like.

Having looked at type-shifting rules and silent operators, we now turn to option (ii), i.e. assuming that all intersective adjectives have translations of a single type, and eliminating type mismatches via a new composition rule. Again for concreteness, we will take that type to be $\langle e,t\rangle$. This means that we need to address the type mismatch that occurs in attributive positions such as *reasonable doubt*, as we saw in (25). Our new rule is called Predicate Modification, though 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 4. 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 $\langle e,t\rangle$

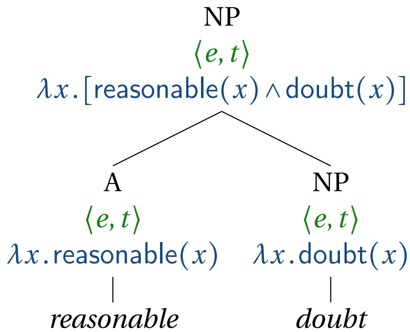
Then:

$$\gamma \rightsquigarrow \lambda u.[\alpha'(u) \wedge \beta'(u)]$$

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*:

(32)



With this in place, the rest of the derivation proceeds as before.

Exercise 3. Consider the sentence *John is a vegetarian farmer*. Give two different analyses of the sentence, one using the Predicate-to-modifier shift, and one using Predicate Modification. Give your analysis in the form of a tree that shows for each node, the syntactic category, the type, and a fully beta-reduced translation. (Feel free to use the Lambda Calculator for this.)

Exercise 4. Identify the types of the following expressions:

(a) $\lambda x \lambda y.\text{in}(y,x)$

(b) $\lambda x.x$

(c) $\lambda x.\text{city}(x)$

(d) texas

(e) $\lambda y\,.\,\mathsf{in}(y, \mathsf{texas})$

(f) $\lambda f\,.\,f$

(g) $\lambda y \lambda x\,.\,\mathsf{Fond\text{-}of}(x, y)$

Assume:

- x and y are variables of type e, and f is a variable of type $\langle e, t\rangle$.
- Any constant that appears with an argument list of length 1 (e.g. city) is a unary predicate, and any constant that appears with an argument list of length 2 (e.g. in) is a (Curried) binary predicate.
- Any constant that appears without an argument list (e.g. texas) is type e.

The following exercises are adapted from Heim & Kratzer (1998).

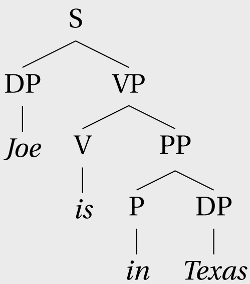
Exercise 5. In addition to the ones given above, adopt the following lexical entries, using the same assumptions about types as in the previous exercise:

1. *cat* $\rightsquigarrow \lambda x\,.\,\mathsf{cat}(x)$
2. *city* $\rightsquigarrow \lambda x\,.\,\mathsf{city}(x)$
3. *gray* $\rightsquigarrow \lambda x\,.\,\mathsf{Gray}(x)$
4. *gray*$_2$ $\rightsquigarrow \lambda P \lambda x\,.\,\mathsf{Gray}(x) \wedge P(x)$
5. *in* $\rightsquigarrow \lambda y \lambda x\,.\,\mathsf{in}(x, y)$

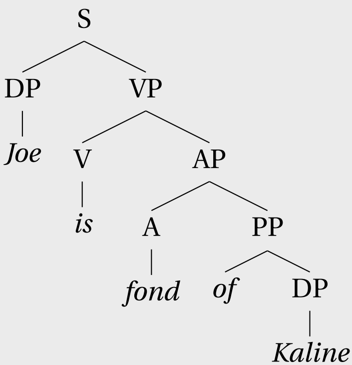
6. *in*$_2 \rightsquigarrow \lambda y \lambda P \lambda x . P(x) \wedge \mathsf{in}(x, y)$
7. *fond* $\rightsquigarrow \lambda y \lambda x . \mathsf{fondOf}(x, y)$
8. *fond*$_2 \rightsquigarrow \lambda y \lambda P \lambda x . P(x) \wedge \mathsf{fondOf}(x, y)$
9. *Joe* $\rightsquigarrow$ joe
10. *Texas* $\rightsquigarrow$ texas
11. *Kaline* $\rightsquigarrow$ kaline
12. *Lockhart* $\rightsquigarrow$ lockhart

For each of the trees below, provide a fully beta-reduced translation at each node, and state the type of the expression.

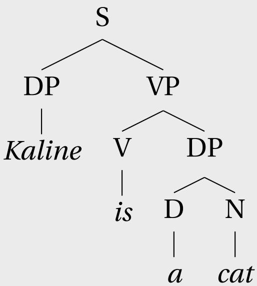
(a)



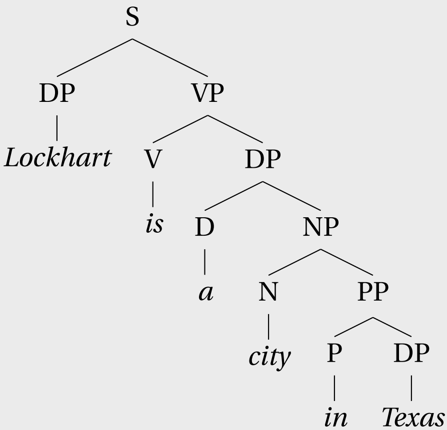
(b)



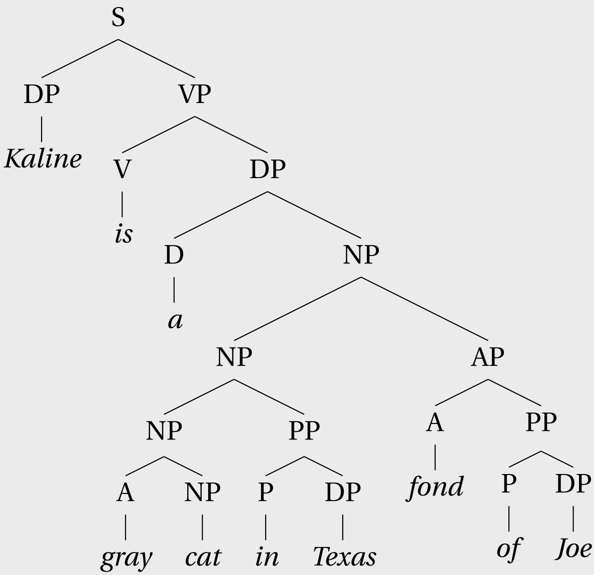
(c)


S
DP
VP
Kaline
V
DP
is
D
N
a
cat

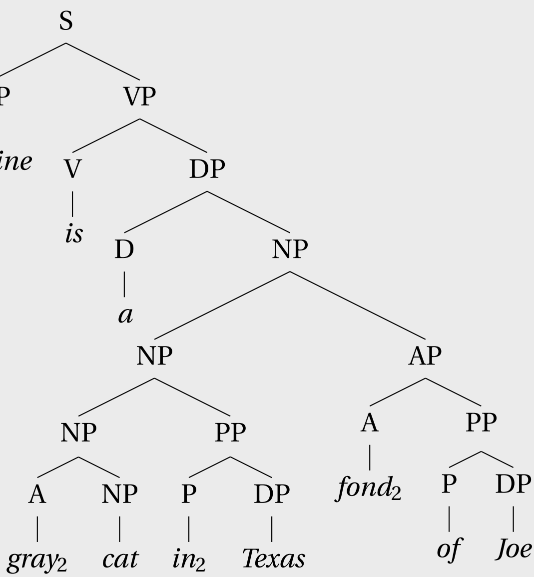
(d)


S
DP
VP
Lockhart
V
DP
is
D
NP
a
N
PP
city
P
DP
in
Texas

(e)


S
DP
VP
Kaline
V
DP
is
D
NP
a
NP
AP
NP
PP
A
PP
A
NP
P
DP
fond
P
DP
gray
cat
in
Texas
of
Joe

(f)


S
DP
VP
Kaline
V
DP
is
D
NP
a
NP
AP
NP
PP
A
PP
A
NP
P
DP
fond₂
P
DP
gray₂
cat
in₂
Texas
of
Joe

Exercise 6. *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*

(b) *presumed*

(c) *future*

(d) *good*

(e) *mere*

Exercise 7. In Russian, there is a morphological alternation between two forms of adjectives, a LONG FORM and a SHORT FORM. For example, the short form of the adjective 'good' is *xoroša* (feminine) or *xoroš* (masculine), and the long form is *xorošaja* (feminine) or *xorošij* (masculine). As discussed by Siegel (1976), the two forms have different syntactic distributions. In attributive positions (modifying a noun), only the long form is possible:

(33) a. Èto byla xoroša-ja teorija.
this was good-LONG theory
'This was a good theory.'

b. *Èto byla xoroša teorija.
this was good.SHORT theory

But in predicative positions, both forms are possible:

(34) a. Èta teorija byla xoroša-ja.
this theory was good-LONG
'This theory was good.'
b. Èta teorija byla xoroša.
this theory was good.SHORT
'This theory was good.'

(35) a. Naša molodež' talantliva-ja i trudoljubiva-ja.
our youth talented-LONG and industrious-LONG
'Our youth are talented and industrious.'
b. Naša molodež' talantliva i
our youth talented.SHORT and
trudoljubiva.
industrious.SHORT
'Our youth is talented and industrious.'

Construct an analysis (including lexical entries, any type-shifting rules you wish to assume, syntactic rules, and perhaps additional constraints) that accounts for this contrast. You may wish to include a lexical entry for the -LONG suffix and/or the -SHORT suffix. Provide derivation trees for each of the grammatical sentences provided in this exercise, and explain why the ungrammatical sentence is ruled out.

7.3 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 (36a), which involves an attributive use of the adjective *reasonable*, in terms of (36b), which involves a predicative use of the same adjective:

(36) 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:

(37) woman who Björn loves

This expression characterizes any individual who has the following two properties: (i) she is a woman; (ii) Björn 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 Björn loves. Such an interpretation can be derived compositionally if we assume that the relative clause *who Björn 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 Björn loves* translate to expressions of type $\langle e,t\rangle$, they can combine via Predicate Modification, like so:

$$\lambda x.[\mathsf{woman}(x)\wedge\mathsf{loves}(\mathsf{b},x)]$$
$$\langle e,t\rangle$$

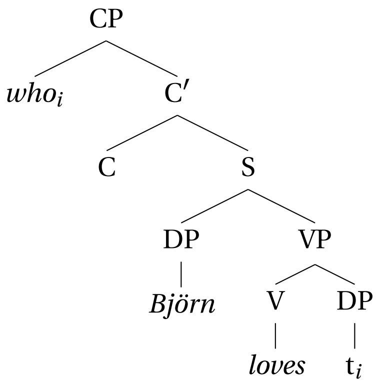
$$\lambda x.\mathsf{woman}(x) \qquad\qquad \lambda x.\mathsf{loves}(\mathsf{b},x)$$
$$\langle e,t\rangle \qquad\qquad\qquad\qquad \langle e,t\rangle$$

woman *who Björn loves*

This expression captures the fact that a *woman who Björn loves* is both a woman and an individual loved by Björn.

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 Björn 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′).[2] 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 Björn loves*.[3]

To explain the fact that *that* and *which* cannot co-occur in

[2]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).

[3]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]
b. *good old-fashioned values $[_{\text{CP}}$ 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):

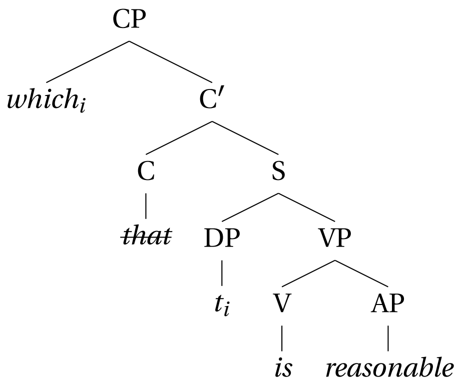
(ii) I woaß ned **wann dass** da Xavea kummt.
I know not **when that** the Xavea comes
'I don't know when Xavea is coming'

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.

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.[4]

The same kind of movement is thought to occur in a relative clause like *who likes Agnetha* 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:

(38)



In this tree, the relative pronoun *which* is co-indexed with a trace in the subject position for the embedded auxiliary verb *is*.[5] 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:

[4]See Carnie (2013, Ch. 12) for a more thorough introduction to the syntax of relative clauses.

[5]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 how to integrate this "copy theory" of movement with compositional semantics, see Pasternak (2020).

- 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. 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.[6]

We have thus arrived at a new composition rule:

[6]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

Composition Rule 5. Pronouns and Traces Rule
If α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow x_i$

We have called it the 'Pronouns and Traces Rule' because it will also be used for pronouns; for example:

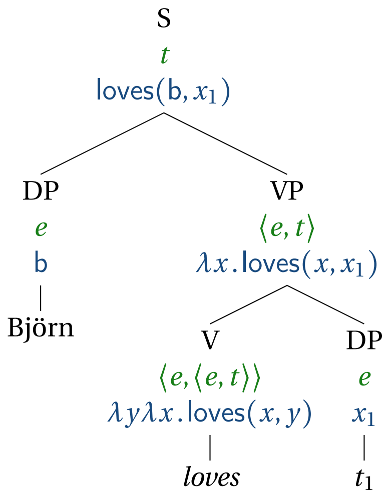
$$he_7 \rightsquigarrow x_7$$

We will see more on the pronoun side of this in Section 7.5.[7]

With these assumptions, we derive the representation

$$\mathsf{loves}(\mathsf{b}, x_1)$$

for *Björn loves* t_1:



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.

[7]The idea of treating traces and pronouns as variables is rather controversial; see Jacobson (1999) and Jacobson (2000) for critique and alternatives.

The translation corresponding to this S node, $\text{loves}(b, x_1)$, 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 Björn 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 $\lambda x\,.\,\text{loves}(b, x)$?

We can achieve this by assigning the relative clause an interpretation in which a lambda operator binds the variable x_1, thus:

$$\lambda x_1\,.\,\text{loves}(b, x_1)$$

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 Lambda Abstraction or Functional Abstraction), triggered by the presence of an indexed relative pronoun, turns the appropriate variable from a free one into a bound one:

Composition Rule 6. Predicate Abstraction
If

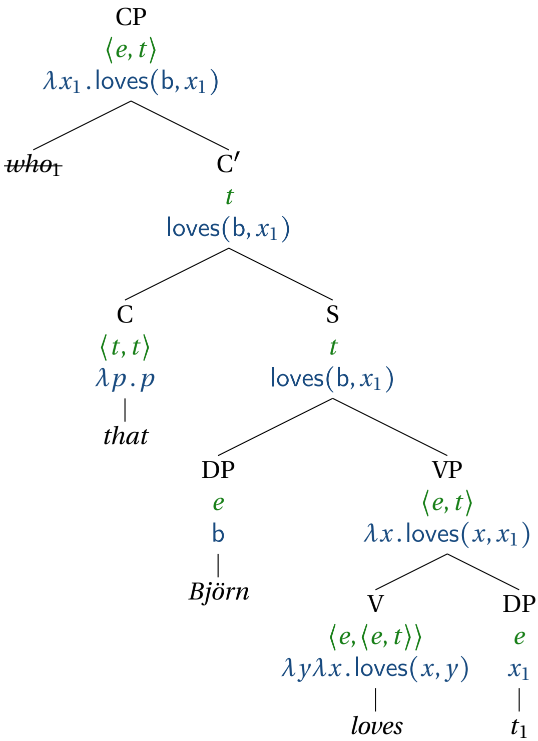
- γ is a syntax tree whose only two subtrees are α_i and β
- α_i is a terminal node carrying the index i
- $\beta \rightsquigarrow \beta'$
- β' is an expression of any type

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 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:

(39)

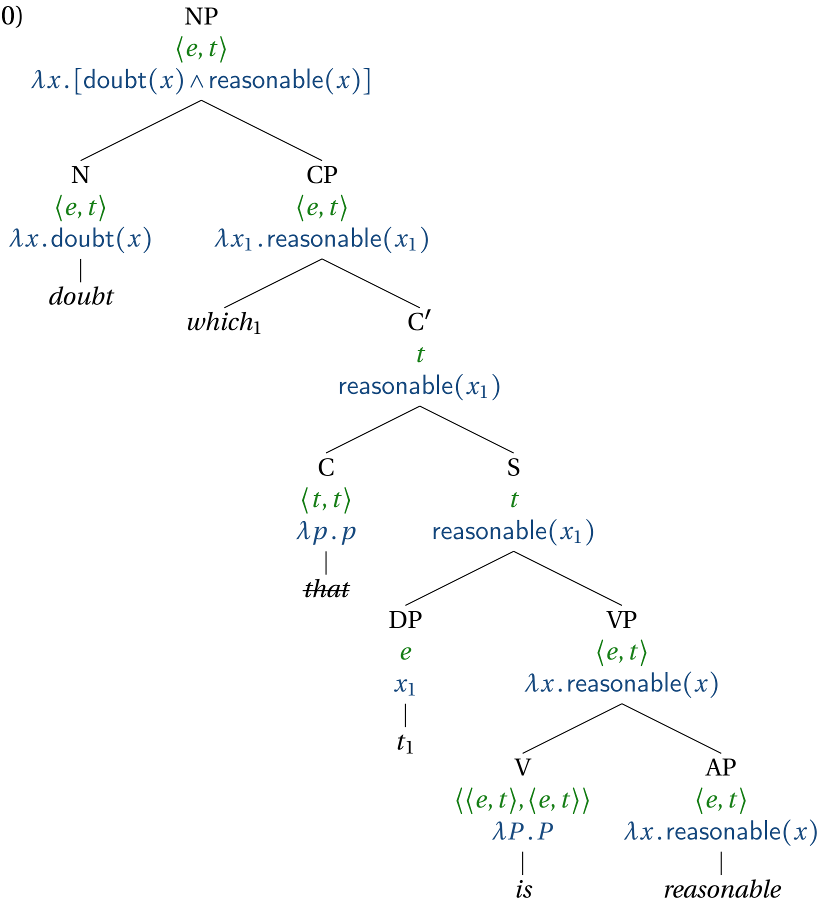


We have reached our goal! The relative clause *that Björn loves* denotes the property of being loved by Björn. Because it translates to an expression of type $\langle e, t\rangle$, it can combine via Predicate Modification with *woman*, giving the property of being in the intersection between the set of women and the set of individuals who Björn 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 Björn, 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 Björn. 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:

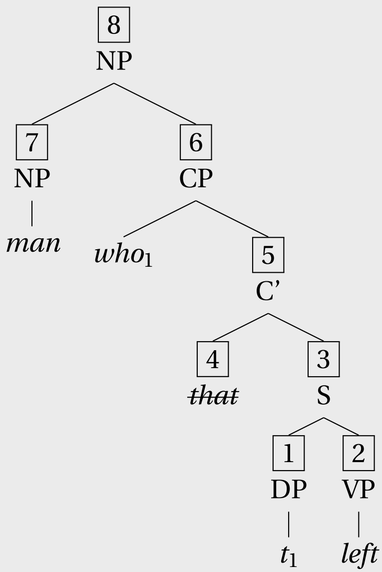
(40)



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 (39) for *the woman that Björn loves, or not, as in* the woman Björn 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 8.
(a) For each of the labelled nodes in the following tree, give: i) the type; ii) a fully beta-reduced translation to L_λ, and iii) the composition rule that is used at the node.



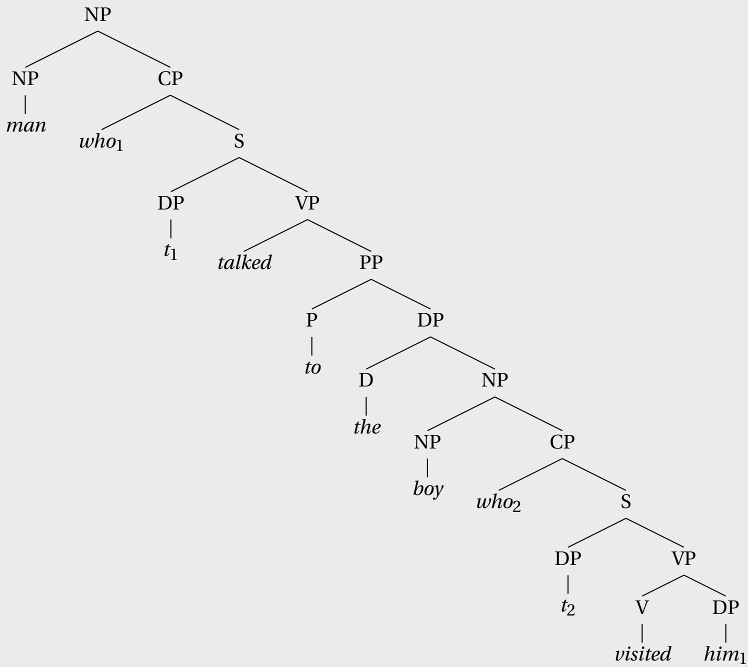
(b) You are not asked to give a type for *who*$_1$. Why not? Hint: Use the word 'syncategorematic'.

Exercise 9. Traditional grammar distinguishes between *restrictive* and *non-restrictive* relative clauses. Non-restrictive relative clauses are normally set off by commas in English, and they can modify proper names and other individual-denoting expressions.

1. Susan, who I like, is coming to the party.
2. *Susan who I like is coming to the party.
3. That woman, who I like, is coming to the party.
4. The woman who I like is coming to the party.

We have given a treatment of restrictive relative clauses in terms of Predicate Modification. Would an analysis using Predicate Modification in the same way be appropriate for non-restrictive relative clauses? Why or why not?

Exercise 10. For each node in the following tree, give the type and a fully beta-reduced translation to L_λ.



You'll need to make an assumption about the denotation of the definite article *the*. For the purposes of this exercise, please assume that it is translated as follows:

$$\textit{the} \rightsquigarrow \lambda P\,.\,\iota x\,.\,P(x)$$

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.

7.4 Quantifiers in object position

7.4.1 Quantifier raising

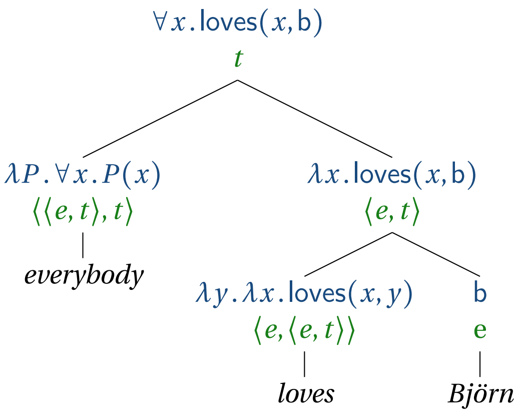
Everybody loves Björn should be translated as:

(41) $\forall x . \mathsf{loves}(x, \mathsf{b})$

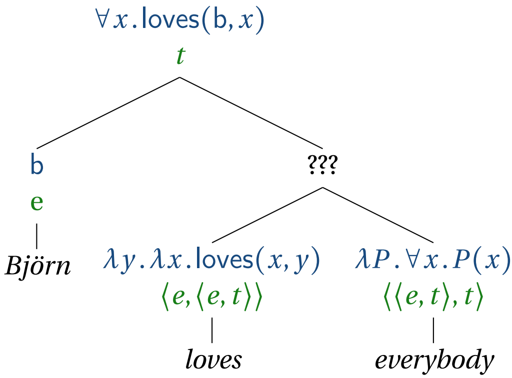
and *Björn loves everybody* should be translated as:

(42) $\forall x . \mathsf{loves}(\mathsf{b}, x)$

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 (*Björn 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. It is rather an embarrassment that this does not work. It is clear what this sentence means!

According to the assumptions we made so far, *everybody* translates as:

(43) $\lambda P.\forall x.P(x)$

The appropriate value for P here would be a function that holds of an individual if Björn loves that individual:

(44) $\lambda x.\mathsf{loves}(\mathsf{b}, x)$

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:

(45) $[\lambda P\forall x.P(x)](\lambda x.\mathsf{loves}(\mathsf{b}, x))$

This beta-reduces to:

(46) $\forall x.\mathsf{loves}(\mathsf{b}, x)$

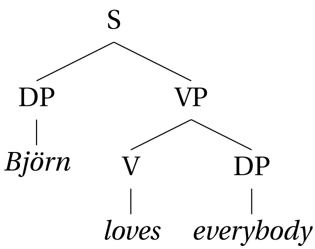
Exercise 11. Before we continue working through the problem raised by *Björn loves everybody*, check your understanding by simplifying the following expression step-by-step:

$$[\lambda Q\,.\,\forall x[\mathsf{Linguist}(x) \to Q(x)]](\lambda x_1\,.\,\mathsf{Offended}(\mathsf{j}, x_1))$$

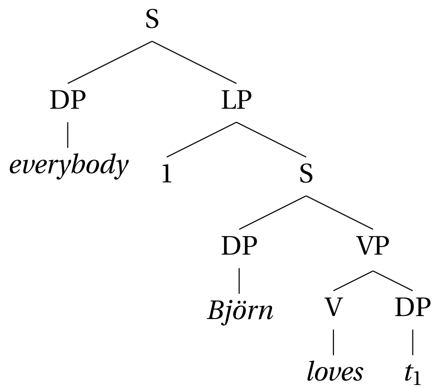
Tip: Use the 'scratch pad' function in the Lambda Calculator.

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 1970's 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 (47a) is converted to the Logical Form representation (47b):

(47) a.



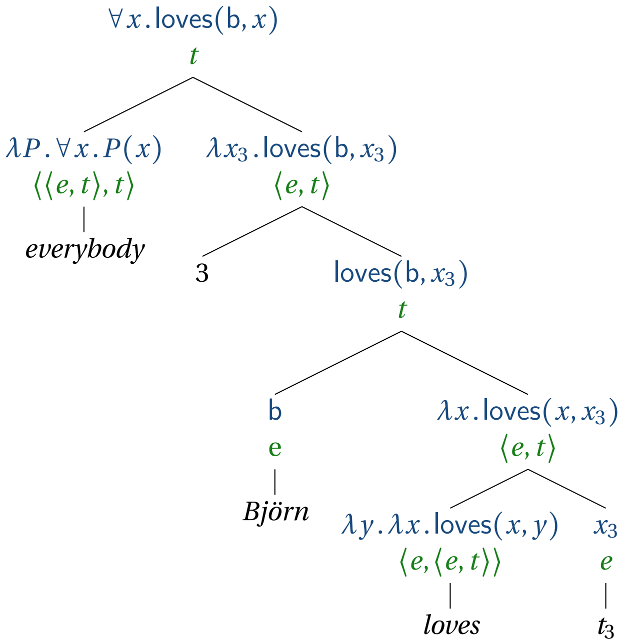
b.



The index 1 in the syntax tree plays the same role as a relative pronoun like *which* in a relative clause: It triggers the introduction of 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 LP for 'lambda P'; Function Application is used at all other branching nodes. The LP 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. LP was introduced by Heim & Kratzer (1998) and has been widely adopted, though the name we use is specific to our textbook.

(48)



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)[8]

Transformations map from DS to SS, and from SS to PF and LF:

$$\begin{array}{ccc} & \text{DS} & \\ & | & \\ & \text{SS} & \\ \swarrow & & \searrow \\ \text{LF} & & \text{PF} \end{array}$$

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.

[8]'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 12. 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 $\langle e, \langle e, t\rangle\rangle$.

Exercise 13. *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.

Exercise 14. Provide a fragment of English with which you can derive truth conditions for the following sentences:

1. Every conservative congressman smokes.
2. No congressman who smokes dislikes Susan.
3. Susan respects no congressman who smokes.
4. Susan dislikes every congressman.
5. Some congressman from every state smokes.
6. Every congressman respects himself.

The fragment should include:

- a set of syntax rules

- lexical entries (translations of all of the words into L_λ)
- composition rules (Function Application, Predicate Modification, Predicate Abstraction, Pronouns and Traces Rule, Non-branching Nodes)

Then, for each sentence:

- draw the syntactic tree for the sentence
- for each node of the syntactic tree:
  - indicate the semantic type
  - give a fully beta-reduced representation of the denotation in L_λ
  - specify the composition rule that you used to compute it

If the sentence is ambiguous, give multiple analyses, one for each reading.

You can use the Lambda Calculator for this exercise.

7.4.2 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 priori* the simplest hypothesis and defends it against putative empirical arguments against it.

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".

Exercise 15. What is the problem of quantifiers in object position, and what are the main approaches to solving it? Explain in your own words.

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, called OBJECT RAISING, defined as follows. 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.

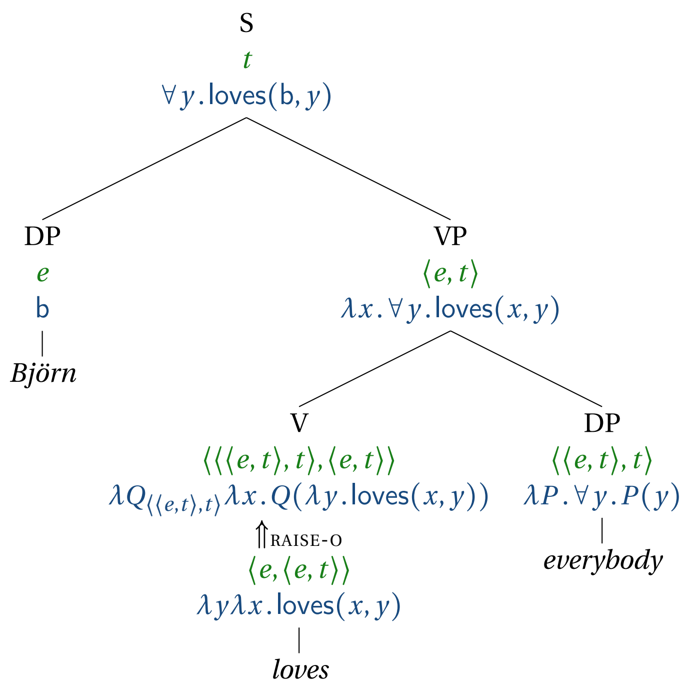
Type-Shifting Rule 2. Object raising (RAISE-O)
If an English expression α is translated into a logical expression α' of type $\langle e,\langle a,t\rangle\rangle$, for any type a, then α also has a translation of type $\langle\langle\langle e,t\rangle,t\rangle,\langle a,t\rangle\rangle$ of the following form:

$$\lambda Q_{\langle\langle e,t\rangle,t\rangle}\lambda x_a\,.\,Q(\lambda y\,.\,\alpha'(y)(x))$$

(unless Q, y or z occurs in α'; in that case, use different variables).

Using this rule, a sentence like *Björn loves everybody* can be analyzed as follows, without quantifier raising:

(49)



The translation of the VP node can be computed from those of the V and object DP nodes through three successive beta reductions:

(50)
$$\begin{aligned}
&[\lambda Q.\lambda x.Q(\lambda y.\mathsf{loves}(x,y))](\lambda P.\forall y.P(y))\\
&\equiv \lambda x.[[\lambda P.\forall y.P(y)](\lambda y.\mathsf{loves}(x,y))]\\
&\equiv \lambda x.\forall y.[[\lambda y.\mathsf{loves}(x,y)](y)]\\
&\equiv \forall y.\mathsf{loves}(x,y)
\end{aligned}$$

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.

(51)

- t — $\exists x \forall y.\mathsf{loves}(x,y)$
  - $\langle\langle e,t\rangle,t\rangle$ — $\lambda P.\exists x.P(x)$
    - *Somebody*
  - $\langle e,t\rangle$ — $\lambda x.\forall y.\mathsf{loves}(x,y)$
    - $\langle\langle\langle e,t\rangle,t\rangle,\langle e,t\rangle\rangle$ — $\lambda Q_{\langle\langle e,t\rangle,t\rangle}\lambda x.Q(\lambda y.\mathsf{loves}(x,y))$
      - $\Uparrow$ RAISE-O
      - $\langle e,\langle e,t\rangle\rangle$ — $\lambda y \lambda x.\mathsf{loves}(x,y)$
        - *loves*
    - $\langle\langle e,t\rangle,t\rangle$ — $\lambda P.\forall y.P(y)$
      - *everybody*

But what about the inverse scope reading, in which the object universal takes scope over the subject existential? It turns out that in order to generate this reading we need to lift both arguments of the verb. To do so, we first need to raise the subject, with a rule we will call Subject Raising. We then lift the verb using the Subject Raising and then the Object Raising rule and combine the resulting doubly-lifted verb with its two arguments.

Type-Shifting Rule 3. 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 y, Q or x is free in α'; in that case, use different variables).

This rule is the mirror image of the Object Raising rule above, in the sense that this rule alters the way that a transitive verb combines with its subject argument, while the Object Raising rule alters the way it combines with its object argument.

We are now ready to generate the inverse scope reading of *Somebody likes everybody*. To do so, we apply Subject Raising to the verb, followed by Object Raising:

(52)

- t
 $\forall y \exists x . \mathsf{loves}(x, y)$
  - $\langle\langle e,t\rangle,t\rangle$
 $\lambda P . \exists x . P(x)$
    - *Somebody*
  - $\langle\langle\langle e,t\rangle,t\rangle,t\rangle$
 $\lambda Q_{\langle\langle e,t\rangle,t\rangle} . \forall y . Q(\lambda x . \mathsf{loves}(x, y))$
    - $\langle\langle\langle e,t\rangle,t\rangle,\langle\langle\langle e,t\rangle,t\rangle,t\rangle\rangle$
 $\lambda Q'_{\langle\langle e,t\rangle,t\rangle} \lambda Q_{\langle\langle e,t\rangle,t\rangle} . Q'(\lambda y . Q(\lambda x . \mathsf{loves}(x, y)))$
 $\Uparrow$ RAISE-O
 $\langle e,\langle\langle\langle e,t\rangle,t\rangle,t\rangle\rangle$
 $\lambda y \lambda Q_{\langle\langle e,t\rangle,t\rangle} . Q(\lambda x . \mathsf{loves}(x, y))$
 $\Uparrow$ RAISE-S
 $\langle e,\langle e,t\rangle\rangle$
 $\lambda y \lambda x . \mathsf{loves}(x, y)$
      - *loves*
    - $\langle\langle e,t\rangle,t\rangle$
 $\lambda P . \forall y . P(y)$
      - *everybody*

Exercise 16. What happens if we apply Object Raising to the verb, followed by Subject Raising? Draw a derivation at the same level of detail as the tree in (52). Can the resulting reading also be generated in a simpler way?

In fact, Subject Raising and Object Raising are both instances of a general type-shifting schema that Hendriks defines. (The following explanation is advanced, and the rest of the current subsection 7.4.2 can be skipped. Nothing in the remainder of the book depends on it.) 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 translations of the following form, where $\vec{x}$ and $\vec{z}$ stand for possibly null sequences of arguments of the same length as $\vec{a}$ and $\vec{c}$ respectively:

(53) $\lambda \overrightarrow{x}_{\overrightarrow{a}} \lambda Q_{\langle\langle b,t\rangle,t\rangle} \lambda \overrightarrow{z}_{\overrightarrow{c}} [Q(\lambda y_b[\alpha'(\overrightarrow{x})(y)(\overrightarrow{z})])]$

(unless x, y, z, or Q occur in α'; in that case, just use different variables of the same type).

This schema works in the following way, for a verb α that expects at least one argument, the "targeted argument" as we will call it. In the following examples, this argument will be of type e, but more generally it could be of any type; this is why the schema uses b instead of e. The sequences $\overrightarrow{x}$ and $\overrightarrow{z}$ represent whatever arguments the verb applies to before and after it combines with the targeted argument. Suppose now that a verb has combined with all of the arguments in $\overrightarrow{x}$ and that its next argument is not of the expected type (say e) but rather it is a quantifier Q of type $\langle\langle e,t\rangle,t\rangle$. In that situation, the verb cannot apply to Q; and if there are more arguments coming up, i.e. if $\overrightarrow{z}$ is nonempty (for example, if Q is in object position, $\overrightarrow{z}$ will contain a slot for the subject), Q cannot apply to the verb either. Hendriks' schema adjusts the entry and type of the verb α by replacing e with $\langle\langle e,t\rangle,t\rangle$ so that α can apply to Q. The adjusted entry provides α with all of the arguments in $\overrightarrow{x}$, then with a fresh variable y, and finally with all remaining arguments in $\overrightarrow{z}$ (such as the subject); and finally it abstracts over y and uses the quantifier Q to bind it. This makes sure that the adjusted entry behaves just as the original entry for α would do if the quantifier Q was raised above α and all of its arguments, leaving a trace corresponding to the variable y.

To illustrate, the Object Raising rule above results from applying Hendriks's schema with $\overrightarrow{x}$ and $\overrightarrow{a}$ as null (because the verb does not apply to any arguments before it combines with the object), b as a (corresponding to the type of the object – typically type e), $\overrightarrow{z}$ as z (because after combining with the object, the verb still expects to apply to the subject), and $\overrightarrow{c}$ as e (because the subject is of type e):

(54) $\lambda Q_{\langle\langle a,t\rangle,t\rangle} \lambda z_e [Q(\lambda y_a[\alpha'(y)(z)])]$

To get the Subject Raising rule, we instantiate Hendriks' schema above by setting $\overrightarrow{x}$ to x, $\overrightarrow{a}$ to a, b to e, and z and $\overrightarrow{c}$ to null:

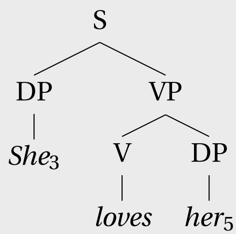
(55) $\lambda x_a \lambda Q_{\langle\langle e,t\rangle,t\rangle}[Q(\lambda y_e[\alpha'(x)(y)])]$

These formulas are identical to those in the Object Raising and Subject Raising rules above, except that we have renamed some bound variables for consistency with the rest of the book.

7.5 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 will try and justify this assumption.

Exercise 17. Using the Pronouns and Traces Rule, give translations at every node for the following tree (ignoring the semantic contribution of gender):



Can all pronouns be interpreted as variables? For example, if someone were to point to Cruella De Vil, and say:

(56) 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:

(57) 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.

Both the deictic and the anaphoric uses can be accounted for under the following hypothesis (to be revised):

Hypothesis 1. All pronouns refer to whichever individual is most salient at the moment when the pronoun is processed.

(We are setting aside gender and animacy features for the moment.) Individuals can be brought to salience in any number of ways: through pointing, by being visually salient, or by being raised to salience linguistically.

The problem with Hypothesis 1 is that there are some pronouns that don't refer to any individual at all. The following examples all have readings on which it is intuitively quite difficult to answer the question, "Who/what does the pronoun refer to?"

(58) No woman blamed herself.

(59) Neither man thought he was at fault.

(60) Every boy loves his mother.

So not all pronouns are referential. It is sometimes said that *No woman* and *herself* are "coreferential" in (58) but this is strictly speaking a misuse of the term "coreferential", because coreference implies reference.

Exercise 18. Give your own example of a pronoun that could be seen as referential, and your own example of a pronoun that could not be seen as referential.

The pronouns in examples (58)-(60) can be analyzed as bound variables.[9] For example, (58) should be translated as:

(61) $\neg\exists x\,.\,[\mathsf{woman}(x) \wedge \mathsf{Blamed}(x, x)]$

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:

(62) 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 have been ungrammatical; inserting the pronoun *she* rescues the sentence (although perhaps not fully; many speakers find examples like this less than fully acceptable). Pronouns in such configurations are called RESUMPTIVE PRONOUNS. The semantic contribution of a resumptive pronoun is exactly like the semantic contribution of a trace: as a variable that is bound by a lambda operator. Thus

who everyone listens to when she speaks

[9] 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.

denotes the property of being an x such that everyone listens to x when x speaks.

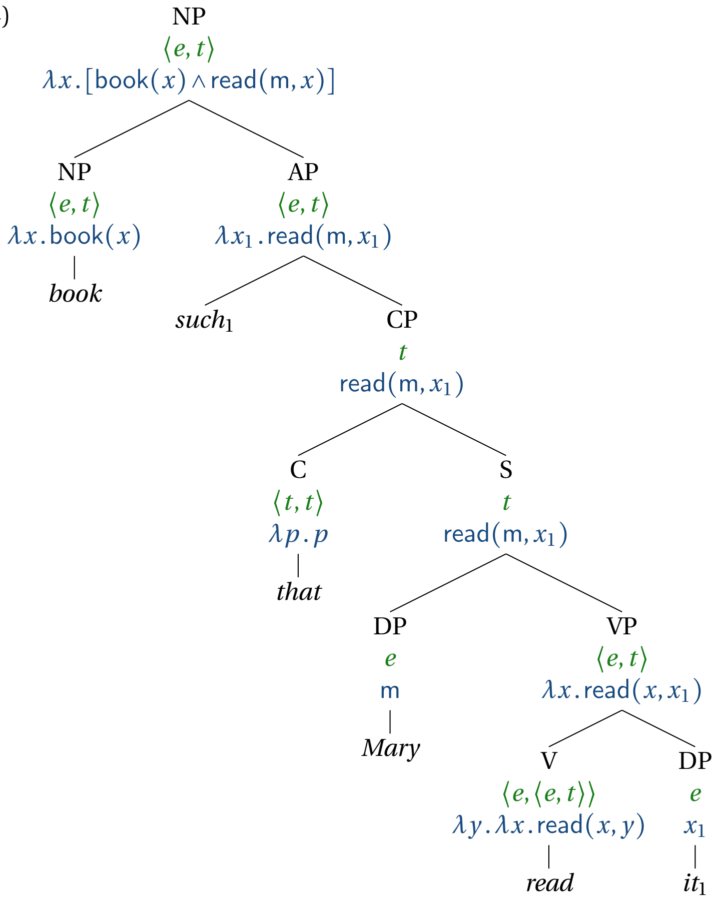
Another example in which pronouns are interpreted very much like traces is with *such*-relatives, as in:

(63) any book such that Mary read it

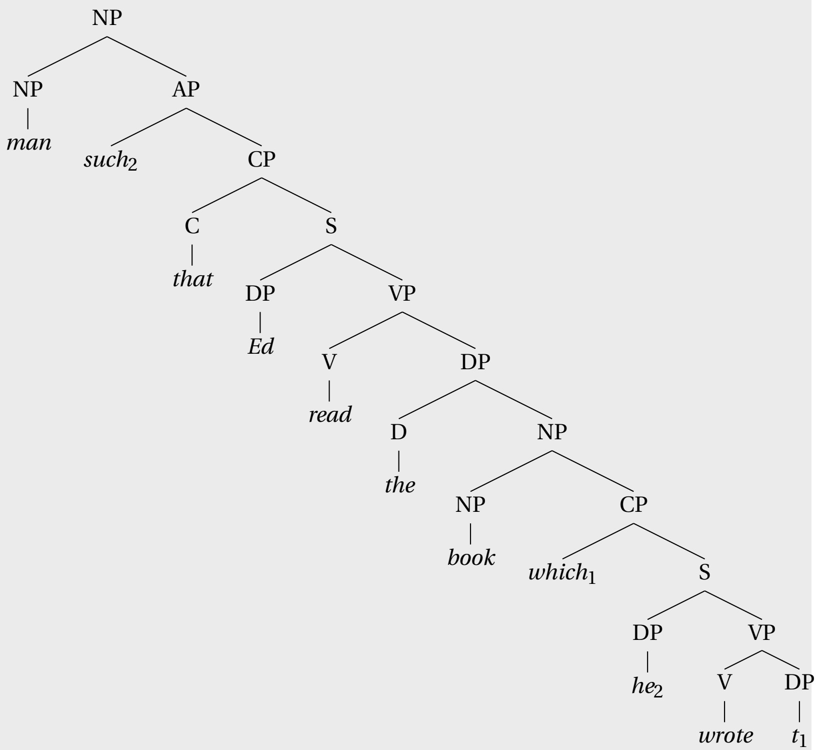
These cases can be treated much like relative clauses, using Predicate Abstraction. The trigger for the abstraction in this case is *such*, which is coindexed with a pronoun rather than a trace. (In this case, the pronoun would not be considered a resumptive pronoun because there's no sense in which it is overtly realizing a trace of movement; *such* is analyzed as originating in its surface position rather than in the position of the pronoun.) For example, in (63), there is coindexation between *such* and *it*. The analysis works as follows:[10]

[10] Keep in mind that x is a distinct variable from x_1.

(64)



Exercise 19. Give the types and a fully beta-reduced logical translation for every node of the following tree (from Heim & Kratzer 1998, p. 114).



For the definite article, assume the analysis described in Exercise 10.

In light of this evidence, let us consider the possibility that pronouns should *always* be treated as bound variables.

Hypothesis 2. All pronouns are translated as bound variables.

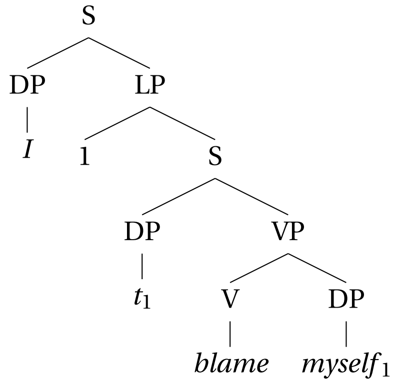
What this means is that whenever a pronoun occurs in a sentence, the sentence translates to a formula in which the variable corresponding to the pronoun is bound by a variable-binder (a lambda or a quantifier).

One reason *not* to treat all pronouns as bound variables is that there are some ambiguities that depend on a distinction between free and bound interpretations. For example, in the movie *Ghostbusters,* there is a scene in which 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 back and they realize that they have never been tested.

(65) 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.

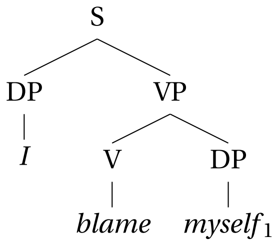
There are two readings of Peter Venkman's quip, a sympathetic reading and a reading on which he is, as usual, being a jerk. On the SLOPPY reading (the sympathetic reading), Peter blames himself. On the STRICT reading (the asshole reading), Peter blames Egon. The strict/sloppy ambiguity exemplified in (65) 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 the property 'x blames x' on the sloppy reading, which is made available through Quantifier Raising thus:

(66)



The strict reading can be derived from an antecedent without Quantifier Raising:

(67)

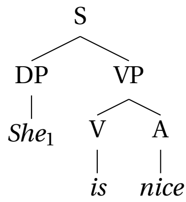


This suggests that pronouns are sometimes bound, and sometimes free. We have not said anything about how to interpret deictic pronouns like *I*, but let us assume that it picks out Peter Venkman just as his name would in the relevant context of utterance. For the reflexive pronoun *myself*, let us assume that it comes with an index that determines which variable it maps to in the representation language, like other pronouns.

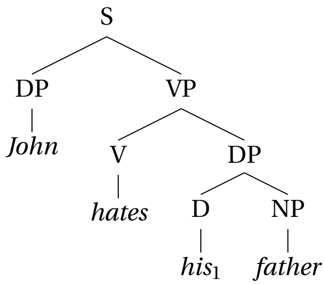
Exercise 20. Which reading — strict or sloppy — involves a bound interpretation of the pronoun? Which reading involves a free interpretation?

These considerations lead us to **Hypothesis 3** (also advocated by Heim & Kratzer (1998)): All pronouns are interpreted as variables, either free or bound. For example, in the following examples, the pronoun in the sentence is interpreted as a free variable; it doesn't end up bound by any quantifier:

(68)

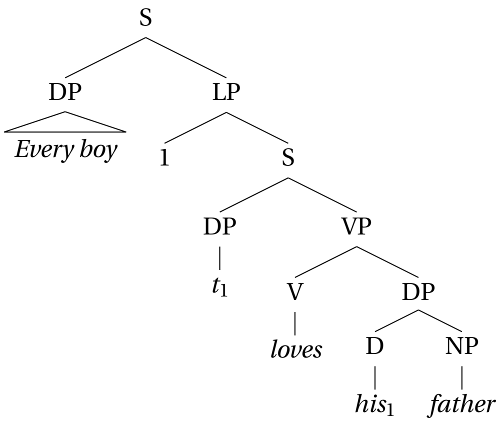


(69)

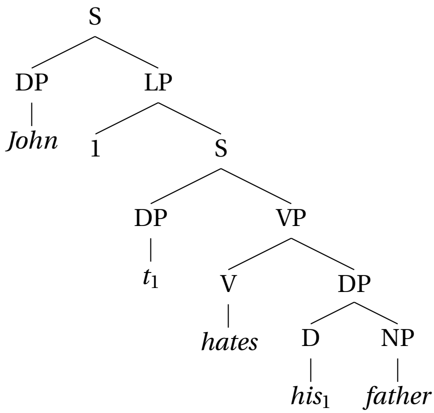


But in the following examples, the pronoun is translated as a bound variable (on the most prominent reading):

(70)



(71)



Whether or not Quantifier Raising takes place will be reflected in a free/bound distinction in the logical translation. The denotation of the sentences with free pronouns will depend on an assignment.

Exercise 21. What empirical advantages does Hypothesis 3 have over Hypotheses 1 and 2? Summarize briefly in your own words, using example sentences where necessary.

This way of treating pronouns suggests that assignment functions can be thought of as being provided by the discourse context. So a sentence that translates to a logical formula containing a free variable can make an interpretable contribution to a discourse. 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.

7.6 Indexicality

So far in this chapter, we have talked about various uses of personal pronouns like *he* and *she*. We modeled both as variables, which can be either free or bound. In the case that they are free, we suggested that their value comes from an assignment function that is determined by context. In this section, we discuss indexical pronouns like *I* and *you*. For these, Kaplan (1977) advocates a slightly different kind of analysis.

An INDEXICAL may be defined as "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*, *that*, *this*, *here*, *now*, *tomorrow*, *yesterday*, *actual*, and *present*. Kaplan distinguishes between two sorts of indexicals:

- DEMONSTRATIVES: indexicals that require an associated demonstration. Examples: *this* and *that*.
- PURE INDEXICALS: indexicals for which no demonstration is required. Examples: *I*, *now*, *here*, *tomorrow* (although *here* has a demonstrative use: "In two weeks, I will be here [pointing]".[11]

Kaplan's logic provides a mechanism for interpreting pure indexicals. It's very simple: Along with a model and an assignment function, Kaplan proposes to interpret logical expressions relative to a context of uutterance.

The CONTEXT OF UTTERANCE determines who is speaking, to whom, when, where, and in what possible world.

$$c = \langle sp, ad, t, loc, w \rangle$$

[11] There is some controversy surrounding how the referents of indexicals are determined: by rules linking expressions to objective features of context, or by speakers' intentions.

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:

$$[\![\phi]\!]^{M,g,c} = \ldots$$

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.

In Kaplan's 'logic of indexicals', there are certain special indexical constants, whose semantics are defined in terms of the context of utterance:

(72)
a. $[\![\textsf{i}]\!]^{M,g,c} = sp(c)$
b. $[\![\textsf{u}]\!]^{M,g,c} = ad(c)$
c. $[\![\textsf{now}]\!]^{M,g,c} = t(c)$
d. $[\![\textsf{here}]\!]^{M,g,c} = loc(c)$

English pure indexicals can then be mapped to these special indexical constants like so:

(73)
a. *I* $\rightsquigarrow$ i
b. *you* $\rightsquigarrow$ u
c. *now* $\rightsquigarrow$ now
d. *here* $\rightsquigarrow$ here

Hence these pure indexical terms get their denotation from the context of utterance, rather than an assignment function. 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.

Kaplan's motivations for proposing this analysis had to do with sentences like *It is necessary that I am here now*, which is arguably not true in most cases (if not all), even though *I am here now* is arguably 'always true' in some sense – true whenever uttered. We will return to the interaction between indexicals and necessity modals in the final chapter, where we move from an extensional semantics to intensional semantics. But it is not just in relation to intensional semantics that indexicality plays a role, and we will encounter several examples of this in the following chapters.

8 Presupposition

8.1 Introduction

There are no dubstep albums by Gottlob Frege (the logician who lived in the 1800s); 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, however it is formulated:

(4) a. Not every dubstep album by Frege is famous.
    b. It's not the case the every dubstep album by Frege is famous.

Thus neither the original sentence nor its negation is felt to be true. How can this be?

The answer is that *every* presupposes the existence of something satisfying the description it combines with. This presupposition is inherited by the negation. 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 A, the negation of A, a yes/no question targeting A, and a conditional sentence in which A figures as the antecedent will all presuppose B as well. Observe that the following sentences also imply that Frege made at least one dubstep album:

(5) Maybe every dubstep album by Frege is famous.

(6) If every dubstep album by Frege is famous, then I must be out of the loop.

Every one of these sentences shares the implication; this is characteristic of presupposition. In general, sentences that embed the original sentence under negation, conditionals, and modals are usually used to test for presuppositions. This is called the FAMILY-OF-SENTENCES TEST. Sometimes questions are also used, though these require an extension of the notion of entailment.

One way of thinking about presupposition is as something that speakers do. For example, someone who speaks of *every dubstep album* presupposes that Frege made at least one dubstep album. 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 can be derived from the speaker-based notion of presupposition as follows: A sentence A presupposes a sentence B if uttering A in any given context acts as a signal that the speaker in that context presupposes B.

We have just given a *pragmatic* characterization of presupposition. Alternatively, presupposition can be given a *semantic definition.* According to the semantic definition, when a sentence presupposes something, the presupposed content must be true in order for the sentence to be true *or* false; otherwise, the sentence just doesn't make any sense. For example, since (1) is not true, (3) is arguably neither true *nor* false. It's just nonsense, because it presupposes something false. As Karttunen (1973b, 170) writes, "There is no conflict between the semantic and the pragmatic concepts of presupposition. They are related, albeit different notions."

The part of the sentence (word or construction) that carries this signal that something is being presupposed is called a PRESUPPOSITION TRIGGER. The word *every* triggers a presupposition of existence. Another example of a presupposition trigger is the adverb *still,* in the sense "up to and including the present". For example, if I said (7), I would signal (8) through a presupposition.

(7) Natalie Portman still speaks French.

(8) 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 (7) is true, (8) 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 (7) as follows:

(9) 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.

(10) 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:

(11) 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,

(12) Does Natalie Portman speak French?

they would not be implying that Natalie Portman spoke French, of course. And yet:

(13) 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.

Here is an example. Example (14a) implies (14b), broadly speaking; anyone who heard (14a) would certainly conclude that (14b) is true, assuming they trusted the speaker.

(14) 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 (14a) 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:

(15) 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.

The projection test does not require the projected inference to have the same properties as an ordinary entailment. Sometimes, projecting presuppositions are defeasible. For example, the following example sounds fine to some native speakers:

(16) Kim's twin sister doesn't live in Austin, because she doesn't have a twin sister.

We will talk about this phenomenon in Chapter 8 under the heading "accommodation". What matters for the projection test is that presuppositions remain present in embedded environments such as negation, whether or not they survive only in a defeasible way.

The decision procedure for distinguishing between the various types of implication relations is summarized in Figure 8.1. Use the defeasibility test to distinguish between entailment and implicature; use the projection test to distinguish between ordinary entailment and presupposition.

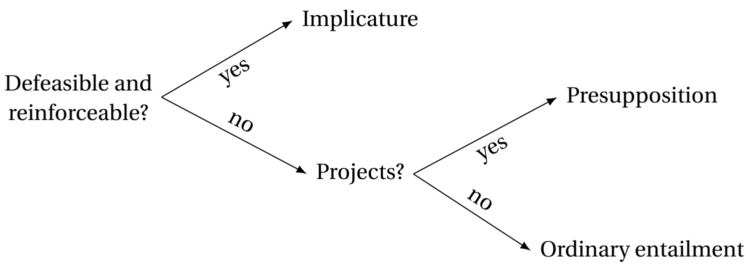


Figure 8.1: A decision tree for categorizing implications

Exercise 1. Use the projection test to determine whether the following implications are entailments or presuppositions. Explain how the test supports your conclusion.

(a) The flying saucer came again.
The flying saucer has come sometime in the past.

(b) The flying saucer came yesterday.
The flying saucer has come sometime in the past.

Exercise 2. Consider the following two sentences.

(17) 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 (18a), but the *succeed* sentence implies that he did (18b), and the *fail* sentence implies that he did not (18c).

(18) 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:

(19) a. (17a) 'succeed' → (18a) 'try';
b. (17b) 'fail' → (18a) 'try';
c. (17a) 'succeed' → (18b) 'did';
d. (17b) 'fail' → (18c) '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*.

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 $\gg$ signifies 'presupposes'):

(20) a. Neither candidate is qualified.
$\gg$ There are exactly two candidates.
b. Ed is glad we won.
$\gg$ We won.
c. Ed knows we won.
$\gg$ We won.
d. Ed's son is bald.
$\gg$ Ed has a son.
e. Only Ed came.
$\gg$ Ed came.
f. The balcony is lovely.
$\gg$ There is a balcony.

The definite determiner is the presupposition trigger that the theory of presupposition grew up around, so we will spend the next section reviewing that history, using the definite determiner as a focal point.

8.2 The definite determiner

So far, we have seen two types for determiners: $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ for the indefinite determiner *a* in predicative descriptions such as *a singer* in *Agnetha is a singer*; and $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ for other determiners. This section motivates a treatment of definite determiners with yet a third type, namely $\langle\langle e,t\rangle,e\rangle$. In a phrase like *the moon*, called a DEFINITE DESCRIPTION, the singular definite determiner *the* takes as input the predicate *moon*, and returns the unique individual which satisfies that predicate, if there is one. If there is not, then the phrase has an 'undefined' denotation. (We set aside plural definite descriptions like *the stars* until Chapter 10.)

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 only one).

In "On Denoting", Russell (1905) proposes to analyze definite descriptions 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':

(21) $$\exists x . [\text{princess}(x) \land \forall y . [\text{princess}(y) \rightarrow x = y] \land \text{smokes}(x)]$$

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.

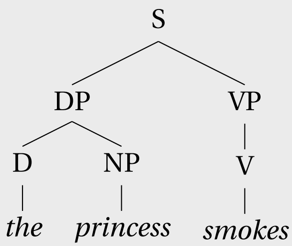
Exercise 3. Read the above formula aloud to yourself and then write out the words that you said. Which part of this formula ensures uniqueness?

Exercise 4.

(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 (21) 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.

(b) What is the type of *the* under your treatment?

(c) Show this lexical entry in action in the following tree:



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").

Putting it another way:[1]

> When a man uses such an expression, he does not *assert,* nor does what he says *entail,* a uniquely existential proposition. But one of the conventional functions of the definite determiner is to act as a *signal* that a unique reference is being made – a signal, not a disguised assertion.

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 <u>ask</u> 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.*

[1] With "disguised assertion", Strawson is alluding to Russell's idea that the form of a sentence containing a definite description, where the definite description appears as a term, is misleading, and that the quantificational nature of definite descriptions is disguised by this form.

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. We do not feel that the sentence is false; we feel that the question of its truth does not arise, as Strawson put it.

For *The king of France is wise*, why doesn't the question of its truth arise? Because the sentence presupposes something that is false, namely that there is one and only one king of France. Only when the presuppositions of a sentence are met can it make enough sense to be true or false. Otherwise, it is *neither true nor false*. In fact, as discussed in Chapter 1, one way of defining presupposition is just in this way:

(22) **Semantic definition of presupposition**
A presupposes B if and only if:
Whenever A is true or false (as opposed to neither true nor false), B is true.

The truth values True and False are called CLASSICAL. The idea here is that a presupposition of a sentence is something that needs to be true in order for the sentence to even have a classical truth value, as opposed to being neither true nor false.

Exercise 5. Recall the definition of entailment:

A entails B if and only if:
Whenever A is true, B is true.

Notice how similar this definition is to the semantic definition of presupposition. Consider the relationship between these two definitions. According to these definitions, is semantic presupposition a species of entailment? Or is it the other way around? Or neither? Explain your reasoning.

One way of implementing the idea that sentences might be neither true nor false is by introducing a third truth value. Under this strategy, along with 'true' and 'false', we have 'undefined' or 'nonsense' as a truth value. Let us use $\#_t$ (pronounced "hash" or "undefined") to represent this undefined truth value. If there is no king of France, then the truth value of the sentence *The king of France is wise* will be $\#_t$. In general, when a sentence has a false presupposition, we call it a PRESUPPOSITION FAILURE. Then the question becomes how we can set up our semantic system so that this is the truth value that gets assigned to a sentence with a false presupposition.

Intuitively, the reason that this sentence is neither true nor false is that there is an attempt to refer to something that does not exist. One way of capturing the same intuition is to introduce a special 'undefined individual' of type e. We will adopt this approach here, using 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.[2] 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

[2] 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'.

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 $\#_e$.

Iota-expressions can be formed with any type; for any variable u, if the type of u is τ, and ϕ is any formula, then $\iota u.\phi$ is an expression of type τ. To make this work, we must assume that every type τ, there is an undefined entity of type τ, written $\#_\tau$. We assume that for any given type τ, the undefined entity $\#_\tau$ is *not* a member of D_τ, the domain of objects of type τ. 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. Following Haug (2013), we can refer to $D_\tau \cup \{\#_\tau\}$ as the FIXED-UP DOMAIN for type τ, and to D_τ as the CLASSICAL DOMAIN for type τ. We denote the fixed-up domain for type τ as D^+_τ. Under the system we define here, every expression of type τ is a member of the fixed-up domain for type τ, but the classical domains play a role in the theory as well.

To add the ι symbol to our logic, first we add a syntax rule producing iota-expressions:

Syntax rule: Iota
If ϕ is an expression of type t, and u is a variable of type τ, then $\iota u.\phi$ is an expression of type τ.

The semantics of iota-expressions is defined as follows:

Semantic rule: Iota
If ϕ is an expression of type t, and u is a variable of type τ,

$$[\![\iota u . \phi]\!]^{M,g} = \begin{cases} d & \text{if } [\![\phi]\!]^{M,g[u \mapsto d]} = \mathsf{T} \text{ but} \\ & \text{for all } d' \in D_\tau \text{ distinct from } d, [\![\phi]\!]^{M,g[u \mapsto d']} = \mathsf{F} \\ \#_\tau & \text{otherwise} \end{cases}$$

Here, d and d' are meta-variables that range over individuals in the classical domain D_τ. The semantic rule tells us that the ι operator picks out the unique value d for the variable u that verifies the condition ϕ. If there is no such value, it returns $\#_\tau$. In essence, ι encodes the existence and uniqueness conditions, and introduces $\#_\tau$ 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 individuals in the model satisfying the scope formula ϕ.

Exercise 6. Read the semantic rule for ι aloud to yourself and then write down the words that you said. How does this definition ensure that ι expressions are undefined when existence and uniqueness are not satisfied?

With this formal tool in hand, we can now give a so-called "Fregean" analysis of the definite determiner as follows (see below for further discussion of why it is called "Fregean"):

(23) *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.

(24)

$$
\begin{array}{c}
\text{DP} \\ e \\ \iota x . \mathsf{moon}(x) \\
\begin{array}{cc}
\text{D} & \text{NP} \\
\langle\langle e,t\rangle,e\rangle & \langle e,t\rangle \\
\lambda P . \iota x . P(x) & \lambda x . \mathsf{moon}(x) \\
| & | \\
\textit{the} & \textit{moon}
\end{array}
\end{array}
$$

(25)
$$
[\![\iota x . \mathsf{moon}(x)]\!]^{M,g} = \begin{cases} d & \text{if } [\![\mathsf{moon}(x)]\!]^{M,g[x \mapsto d]} = \mathsf{T} \text{ but} \\ & \text{for all } d' \in D_e \text{ distinct from } d, \\ & [\![\mathsf{moon}(x)]\!]^{M,g[x \mapsto d']} = \mathsf{F} \\ \#_e & \text{otherwise} \end{cases}
$$

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 True, otherwise False. 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 $\#_e$.

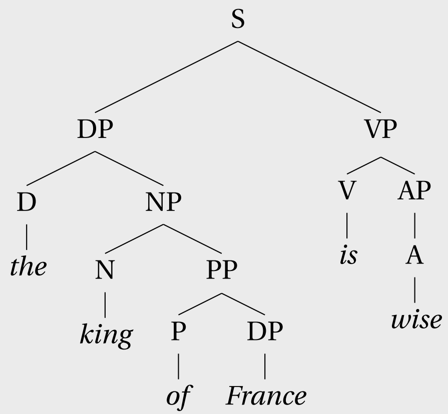
A definite description that successfully refers to something will behave just like a proper name when embedded in a larger sentence. Consider for example:

(26) 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 D_e^+ to D_t^+, and that it maps the undefined individual $\#_e$ to the undefined truth value $\#$. In general, we assume that function-denoting non-logical constants are STRICT in this sense, given by LaPierre (1992): They map an undefined value to an undefined value. Hence, if α denotes $\#_e$, then $\mathsf{wise}(\alpha)$ denotes the undefined truth value $\#$.[3] Since (the translation of) *the king of France*, in a model 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.

Exercise 7. Using the assumptions above, compute a derivation for the following tree:



In the case of *the king of France*, we have a situation where

[3] Let us assume furthermore that if an undefined value $\#_{\langle\sigma,\tau\rangle}$ applies to an argument of type σ, then the result is $\#_\tau$.

the existence presupposition of the Fregean definite article is violated; an expression of the form *the F* where there is no *F*. A situation where the uniqueness presupposition is violated would be a case where there is more than one *F*. For example, there is more than one river in France, so *The river in France is wide* would not have a defined truth value under this analysis.

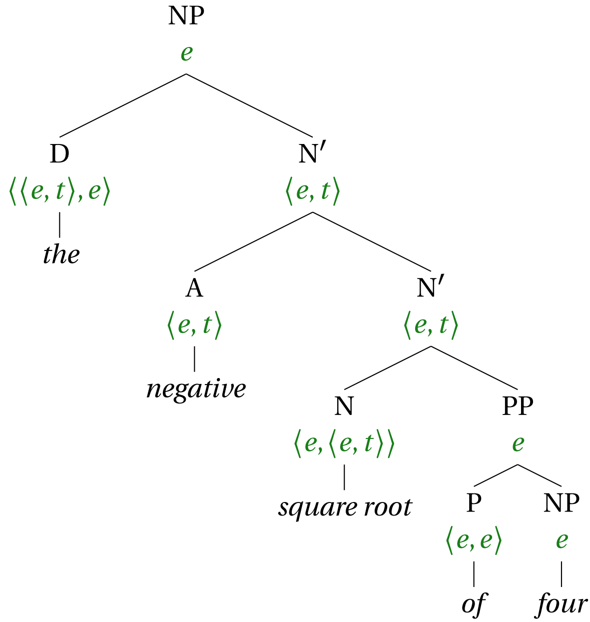
This analysis is called "Fregean", as it captures an intuition that was expressed earlier by Frege (1892 [reprinted 1948]). One passage in which Frege's view on the definite article comes forth involves a discussion of the expression *the negative square root of 4*. According to Frege, such an expression, 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".

To flesh out Frege's analysis of this example further, let us assume that *square root* is a RELATIONAL NOUN, with a denotation of type $\langle e, \langle e, t\rangle\rangle$, following Heim & Kratzer (1998). Let us assume further that *of* makes no contribution to the semantics other than an identity function of type $\langle e, e\rangle$. Spelling this out yields the structure in (27):

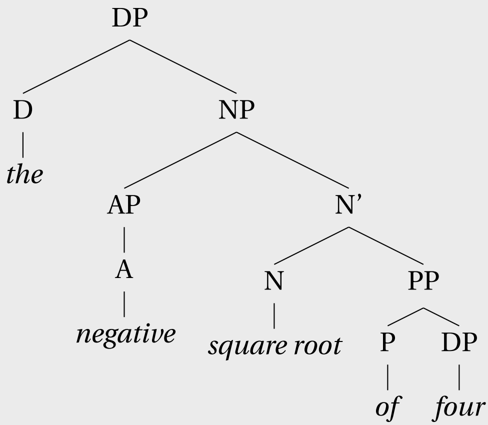
(27)



(The treatment of *square root* as a relational noun is unrelated to Frege's analysis of definite descriptions; we include it in the discussion here only because it figures in the example that Frege uses in the passage quoted above.)

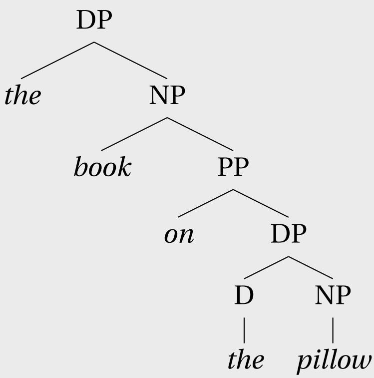
Now, recall that Frege said that the use of a definite description like this is "permissible" only if one and only one object falls under the description (here, being a negative square root of four). What does Frege mean by "permissible"? One way of formalizing this idea is that *the* denotes a function of type $\langle\langle e,t\rangle,e\rangle$ that is only *defined* for input predicates that characterize one single entity. This function applies to a predicate, and if there is exactly one satisfier of that predicate, then the return value is that satisfier. But if there are zero satisfiers or multiple satisfiers, then the function returns a special 'undefined' value.

Exercise 8. 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:

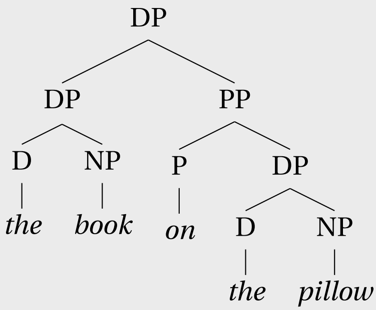


Exercise 9. Consider the two following possible parse trees for *the book on the pillow.*

(i)



(ii)



Assume the following lexical entries:

1. *book* $\leadsto \lambda x.\mathsf{book}(x)$
2. *on* $\leadsto \lambda y \lambda x.\mathsf{on}(x, y)$
3. *pillow* $\leadsto \lambda x.\mathsf{pillow}(x)$

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.

Exercise 10. Explain how this Fregean treatment of the definite determiner vindicates Strawson's intuitions.

Let us consider another example. Beethoven wrote one opera, namely *Fidelio*, but Mozart wrote quite a number of operas. So in a model reflecting this fact of reality, the phrase *the opera by Beethoven* has a defined value. But *the opera by Mozart* does not. Consider what happens when *the opera by Mozart* is embedded in a sentence like the following:

(28) The opera by Mozart is Italian.

This would have the following translation:

$$\mathsf{italian}(\iota x.[\mathsf{opera}(x) \wedge \mathsf{by}(x, \mathsf{mozart})])$$

Assuming that italian (like wise) yields the value $\#$ when applied to an expression whose semantic value is $\#_e$, this formula will denote $\#$ in a model where there are multiple operas by Mozart. Here as before, the undefinedness of the definite description "percolates up", as it were, to the sentence level.

Exercise 11. Both *The king of France is wise* and *The opera by Mozart is Italian* have an undefined value relative to the actual world, but for different reasons. Explain the difference.

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:

(29) Björn loves Agneta's cat.
$\gg$ Agneta has a cat.

The fact that this inference is a presupposition can be seen via the projection test; for example, *Björn doesn't love Agneta's cat* also implies that Agneta has a cat. How might we define possessive-marking in a way that captures this presupposition? Let us consider first what sort of meaning representation we wish to derive for the sentence as a whole. We propose the following as a representation of the meaning of (29):

(30) $\mathsf{loves}(\mathsf{b}, \iota y[\mathsf{cat}(y) \wedge \mathsf{has}(\mathsf{a}, y)])]$

Under this treatment, (29) presupposes not only that Agneta has a cat, but also that she has exactly one (a common but slightly controversial assumption). To arrive at this formula compositionally, we propose the following lexical entry for possessive *'s*:

(31) *'s* $\rightsquigarrow \lambda x \lambda P . \iota y . P(y) \wedge \textsf{has}(x, y)$

Exercise 12. Draw a derivation tree for example (29) and annotate each node with its translation and its semantic type, using the lexical entry in (31).

8.3 Presupposition accommodation

Earlier in this chapter, we said that a *speaker* presupposes some proposition when they take it for granted, treating it as uncontroversial and known to everyone participating in the conversation (in the COMMON GROUND). For example, suppose you work in an office and you have no idea whether your boss Malcolm owns any animals. Then your co-worker tells you:

(32) Malcolm loves his elephant.

You would likely be surprised, and perhaps react with something like *Hey, wait a minute! I didn't know Malcolm owns an elephant.* This is the sort of reaction that might be expected when a sentence presupposes content that is not already in the common ground.

But suppose instead that in the same context, your co-worker tells you:

(33) Malcolm loves his cat.

In this case, you would be more likely to take the information about his cat in stride, add it to your stock of beliefs about Malcolm without raising a fuss. That is to say, you would probably treat (33) as if it meant something like this:

(34) Malcolm owns a cat, and he loves it.

Now, the diagnostics from Chapter 1 show that the implication of (32) that Malcolm has an elephant, and the implication of (33) that Malcolm has a cat, are (semantic) presuppositions of these two sentences. And the system we have described so far would assign them the truth value # if John doesn't own an animal of the required kind. If the *Hey, wait a minute* reaction (cf. von Fintel 2004) is what naturally happens in discourse when a sentence lacks a classical truth value due to its presuppositions not being satisfied in the discourse context, then (32) and (33) should elicit similar responses. But their effects on the discourse are quite different.

PRESUPPOSITION ACCOMMODATION is a process by which hearers update their beliefs with the presupposed content of a sentence after hearing it, so that the sentence can be understood in a revised context where its presuppositions hold in the common ground. It's as if the presupposed content were asserted, silently and just immediately before the sentence, so that the presupposition gets promoted, in effect, to an entailment. Some speakers feel that (34) has no presupposition: in contexts where John doesn't own a cat, it is simply judged false. Through accommodation, the presupposition of (33) has changed from a test whose failure leads to undefinedness (the truth value #) to one whose failure leads to falsehood (the truth value F).

Evidently, some presupposed content is easier to accommodate than others. A speaker can easily slip a pet cat into the common ground without much notice, but a pet elephant is more conspicuous. Plausibility plays a role in accommodation, in other words.

When trying to assess whether some bit of meaning is a presupposition or not, it is important to keep in mind the possibility of accommodation. In the case of definite descriptions, accommodation helps an advocate of the Frege/Strawson analysis to explain why the following example from Russell is not a contradiction:

(35) The king of France isn't bald – there is no king of France!

If the first sentence always had the 'undefined' truth value relative to any circumstance lacking a unique king of France, then these two sentences could never be true at the same time; (35) should be a contradiction. Russell uses this example as evidence in favor of his own treatment of definite descriptions; indeed, here, the existence component of the meaning behaves like an entailment rather than a presupposition. But through accommodation, the first sentence can turn into something equivalent to "It's not the case that there is a king of France and he is bald." A defender of the Frege/Strawson approach to definite descriptions can thus appeal to the mechanism of accommodation in order to explain away the apparent counterexample.

Once accommodation is sanctioned as a theoretical possibility, a much wider range of interpretations is predicted to be available for any given sentence containing a presupposition trigger. In general, more careful argumentation is then required in order to diagnose presuppositions, specifically to distinguish them empirically from entailments.

8.4 Definedness conditions

So far we have seen one mechanism for representing presuppositions: the iota operator (ι). In this section, a second one will introduced: the partial operator (∂). The partial operator is a general formal tool for representing definedness conditions.

Recall that in the previous section, we encountered two examples of presuppositional expressions, namely the definite determiner *the* and the possessive suffix *'s*. We translated both using ι-expressions, which lead to a presupposition failure when nothing satisfies the description. In that case, we assumed that definite descriptions denote a special 'undefined individual', denoted $\#_e$.

Not all presuppositions involve the presupposition of existence for a given referent, though. A more general way of dealing with presupposition is needed. 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:

(36) a. Both candidates are qualified.
b. Neither candidate is qualified.

If there were only two candidates and both were qualified, then (36a) would clearly be true and (36b) 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:

(37) a. It's not true that both candidates are qualified.
b. It's not true that neither candidate is qualified.

(38) 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.

(39) 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 (36b). 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] This is what is pre-

[4] $|P| = 2$ is short for $\exists x \exists y[\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg \exists z[\neg(z = x) \wedge \neg(z = y) \wedge P(z)]]$.

supposed. To signify that it is presupposed, we will use Beaver & Krahmer's (2001) ∂ operator, pronounced "presupposing that". It can be referred to as the 'partial operator'. This operator is of type $\langle t, t\rangle$; that is, it maps a formula to another formula. A formula like this:

$$\partial(|P| = 2)$$

can be read, 'presupposing that there are exactly two Ps'. The lexical entry for *neither* can be stated using the ∂ operator as follows:

(40) *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.

In order to be able to give translations like (40), 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:

$$[\![\partial(\phi)]\!]^{M,g} = \begin{cases} \mathsf{T} \text{ if } [\![\phi]\!]^{M,g} = \mathsf{T} \\ \# \text{ otherwise.} \end{cases}$$

The lexical entry in (40) will give us the following analysis for (36b), where beta-reduced variants of the translations are given at each node:

(41)

- $\partial(|\text{candidate}| = 2) \wedge \neg\exists x . [\text{candidate}(x) \wedge \text{qualified}(x)]$
  - $\lambda Q . [\partial(|\text{candidate}| = 2) \wedge \neg\exists x . [\text{candidate}(x) \wedge Q(x)]]$
    - $\lambda P \lambda Q . [\partial(|P| = 2) \wedge \neg\exists x . [P(x) \wedge Q(x)]]$
      - *neither*
    - $\lambda x . \text{candidate}(x)$
      - *candidate*
  - $\lambda x . \text{qualified}(x)$
    - *is qualified*

The translation for the whole sentence should have a defined value in a model if $|\text{candidate}| = 2$ is true in the model. If it has a defined value, then its value is equal to that of $\neg\exists x . [\text{candidate}(x) \wedge \text{qualified}(x)]$.

Let us consider another example of a presupposition that can be represented using definedness conditions: the existence presupposition of the quantifier *every*. This presupposition can be captured using the following lexical entry for *every*:

(42) *every* $\rightsquigarrow \lambda P \lambda Q . [\partial(\exists x . P(x)) \wedge \forall x . [P(x) \rightarrow Q(x)]]$

This treatment will give 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.

Exercise 13. (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)?

8.5 Designing a three-valued logic

In line with Strawson's intuition that sentences with presupposition failures are neither true nor false, we have treated presupposition failures as cases where an expression lacks a classical truth value. We introduced a third truth value, #, representing 'nonsense', to use as the denotation for sentences containing presupposition failures. We generalized this strategy to every type, so that we have special undefined entities of every type.

An alternative, subtly different strategy would have been to abstain from assigning any denotation whatsoever to empty descriptions and other expressions whose presuppositions are not satisfied. These expressions would then be undefined in a more radical sense, lacking any sort of denotation whatsoever, rather than denoting an undefined entity. This radical type of undefinedness is what results in the Heim & Kratzer (1998) system when a partial function is applied to an argument that is not in its domain.

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). A related issue is as follows: if f is a partial function that is not defined for x, then strictly speaking $f(x)$ makes no sense; it's supposed to denote the result of applying f to x but f cannot be applied to x. 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. If we give the status of object to the undefined for every type, then we can be guaranteed that the representation-language expressions that we derive have denotations, as long as they are well-formed.

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 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 8.1. 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. If, on the other hand, we take unde-

∧	T	F	#
T	T	F	#
F	F	F	#
#	#	#	#

∨	T	F	#
T	T	T	#
F	T	F	#
#	#	#	#

	¬
T	F
F	T
#	#

Table 8.1: Truth tables for the Weak Kleene connectives

∧	T	F	#
T	T	F	#
F	F	F	**F**
#	#	**F**	#

∨	T	F	#
T	T	T	**T**
F	T	F	#
#	**T**	#	#

	¬
T	F
F	T
#	#

Table 8.2: Truth tables for the Strong Kleene connectives

finedness to represent 'unknown value', then the conjunction of an unknown value with False is false, and the disjunction of an unknown value with True is true. These connectives are called the STRONG KLEENE connectives (see Table 8.2; the values that differ from the Weak Kleene ones are bolded).

Strong Kleene and Weak Kleene connectives give different truth conditions in the case where one conjunct/disjunct is undefined and the other is not, such as the following:

(43) The king of France is wise and France is located in Africa.
Weak Kleene: #; Strong Kleene: F

(44) The king of France is wise or France is located in Europe.
Weak Kleene: #; Strong Kleene: T

(These truth values are based on the assumptions that there is no king of France, and that France is not located in Africa but rather in Europe.) Intuitions may differ regarding whether it is more sensible to regard (43) as undefined or false, and whether it is more sensible to regard (44) as undefined or true. In any case, there are

	∂
T	T
F	#
#	#

	A
T	T
F	F
#	F

Table 8.3: Truth tables for the partial operator and the assertion operator

trade-offs.

Without any additional mechanisms, the Strong Kleene connectives give us a bit more flexibility: We can define Weak Kleene connectives in terms of Strong Kleene ones, but not the other way around. Furthermore, as we discuss in greater detail in §8.8, while projection is a defining characteristic of presuppositions, they do not always project; they sometimes get 'filtered' or 'plugged'. On their own, the Weak Kleene connectives seem to predict that presuppositions will always project, while the Strong Kleene connectives cause some presuppositions to be cancelled. These are *prima facie* arguments in favor of the Strong Kleene connectives.

However, a system with Weak Kleene connectives can be augmented with an assertion operator $\mathbf{A}$, a "presupposition wipe-out device" (Beaver & Krahmer, 2001, 167). $\mathbf{A}(\phi)$ is true if ϕ is true, and false otherwise. A truth table for the assertion operator is given in Table 8.3, alongside a truth table for the partial operator ∂ introduced in the previous section. With the assertion operator in the inventory of connectives, the logic has greater expressive capacity, and it becomes possible to capture the disappearance of some presuppositions even in the context of Weak Kleene binary connectives. We'll see an example of how this can work in §8.8.

It is sometimes useful to think of the semantic contribution of a sentence as a conjunction consisting of two components: the presupposition and the AT-ISSUE CONTENT. The at-issue content is the part of a sentence which, intuitively speaking, expresses its "main point". In contrast, the presupposition typically corresponds to a background assumption. For example, the presupposition of (45a) is (45b) and its at-issue content is (45c).

(45) a. John stopped smoking.
b. John used to smoke.
c. John does not currently smoke.

When this simplified picture is implemented using the ∂ operator, one conjoins the presupposition $\partial(\pi)$ with the at-issue content ϕ. If so, it is crucial to use a Weak Kleene conjunction, so that $[\partial(\pi) \wedge \phi]$ denotes the truth value of the at-issue content ϕ whenever ϕ is true, and otherwise #. This is the desired behavior. Under a strong Kleene interpretation, when both ϕ and ψ are false, $[\partial(\phi) \wedge \psi]$ is also false. But this is incorrect, since a sentence whose presupposition is false has the truth value #. In light of this consideration, and in keeping with our understanding of the third truth value as 'nonsense', we will adopt the Weak Kleene connectives here. Nevertheless, we encourage the reader to keep in mind Strong Kleene as an alternative. (An alternative solution to this problem, using the Strong Kleene connectives, would be to use a binary 'transplication' operator that connects a non-presupposed formula to a presupposed one, as Beaver & Krahmer (2001, p. 150) do. Muskens (1995a) gives arguments in favor of such an approach.)

Another slightly thorny issue is 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 our understanding of the third truth value as 'nonsense', and assuming that any equality statement involving the undefined individual 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.

The last remaining issue we must address before we can define a three-valued logic to serve as our representation language is what to do with quantified sentences. Consider the following sentence:

(46) Every boy loves his cat.

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 cat* as synonymous with *the cat he has*), this would be translated:

(47) $\forall x\,.\,[\mathsf{boy}(x) \to \mathsf{loves}(x, \iota y[\mathsf{cat}(y) \wedge \mathsf{has}(x,y)])]$

This formula will be true in a model where every element of D_e satisfies the following formula, when plugged in for x:

(48) $\mathsf{boy}(x) \to \mathsf{loves}(x, \iota y[\mathsf{cat}(y) \wedge \mathsf{has}(x,y)])$

What, precisely, is the presupposition of (46)? The iota expression will have a defined value just in case there is exactly one entity y satisfying the following description:

(49) $[\mathsf{cat}(y) \wedge \mathsf{has}(x,y)]$

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 cats. 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 cat* loves that cat? In other words, if the assortment of truth values that the open formula *x loves his cat* (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 cat* (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 value of the variable, and (b) the scope proposition is false for at least one value of the variable. For example, *Every boy loves his cat* is false only if every boy has exactly one cat and at least one boy doesn't love his cat.

Exercise 14. Consider a model with a boy named Freddie who has no cats, and two other boys. Both of the other boys have exactly one cat that they love. Under the treatment of universal quantification just given, what truth value does (47) have in this model?

Notice that the universal quantifier in this example ranges over the elements of D_τ. Recall from our discussion above that $\#_\tau$ is *not* a member of this set. The quantifier ranges over the ordinary domain D_τ, which excludes the partial object $\#_\tau$, rather than the fixed-up domain $D_\tau \cup \{\#_\tau\}$. Suppose it were otherwise; then universal statements would very frequently turn out to be undefined, due to the fact that the scope formula would be undefined when the variable in question is mapped to the undefined entity. Luckily, though, we have excluded it from the domain of the quantifier.

In 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 cat* is false if every boy has exactly one cat (and no boy loves his cat).

Exercise 15. A famous example in the literature on presupposition is Heim's (1983b) *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.

Our treatment of identity and quantification can help to give us a handle on certain exceptions to the general rule that predications involving the undefined individual will themselves be undefined. Above, we said that a predicate should yield $\#$ when applied to the undefined individual, as in *The king of France is wise.* But there are cases in which we might want the predication to be true. Among them are negative existence statements such as the following famous case, discussed by Russell (1910) in response to writings of Alexius Meinong.

(50) The golden mountain does not exist.

Meinong argued that while the golden mountain doesn't actually exist, it has a secondary quasi-existence that allows us to refer to it. Russell (1905) disagreed heartily with Meinong, but used negative existence statements like this as evidence against Frege's analysis of the definite article, in favor of his own. A Frege-friendly approach to negative existence statements involving definite descriptions like (50) is to translate the English word *exist* as follows:

(51) *exist* $\rightsquigarrow \lambda x \,.\, \exists y \,.\, \mathbf{A}(y = x)$

Here we have used the assertion operator $\mathbf{A}$ so that the equality statement comes out as false rather than undefined when an ordinary individual is compared to $\#_e$. Given the lexical entry in (51), the translation of (50) would be:

(52) $\neg\exists y.\mathbf{A}(y = \iota x.[\mathsf{golden}(x) \wedge \mathsf{mountain}(x)])$

In models where there is no unique golden mountain, the term $\iota x.[\mathsf{golden}(x) \wedge \mathsf{mountain}(x)]$ denotes the undefined individual $\#_e$. Hence any equality statement involving it will be undefined. The assertion operator turns undefined into false. So the following formula:

(53) $\exists y.\mathbf{A}(y = \iota x.[\mathsf{golden}(x) \wedge \mathsf{mountain}(x)])$

denotes F, and the formula (52) denotes T.[5]

8.6 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 term is written at the beginning of the value description of that term, in between a colon and a dot. For example, (40) 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. However, one difference is related to the fact that the Heim and Kratzer

[5] Another exception to the rule that predications involving the undefined individual are themselves undefined comes from sentences in which a definite description that fails to refer occurs in a non-subject position (Strawson, 1964):

(i) The Exhibition was visited yesterday by the king of France.

While this sentence still presupposes the existence of a unique king of France, it is still readily judged false (von Fintel, 2004). We offer no account of this phenomenon.

style does not include undefined truth values as part of the ontology; rather, a function whose domain restrictions are not met by a given candidate input simply cannot apply to it. In other words, the Heim and Kratzer style involves what we referred to as 'radical undefinedness' above. This yields the problem that LaPierre (1992, p. 521) pointed out: If a function f is not defined for input x, then there is no sense in talking about the application of another function f' to $f(x)$; in other words $f'(f(x))$ makes no sense. In this respect, under the Heim and Kratzer system, there is a limitation on one's ability to meaningfully assign semantic representations to arbitrarily complex expressions of English.

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. So while the colon-dot system allows expressions like

$$\lambda x : \mathsf{female}(x) \,.\, x$$

(the identity function restricted to females), we have no direct equivalent in our system. However, the iota-operator allows a workaround for that; we can represent that function as:

$$\lambda x \,.\, \iota y \,.\, [\mathsf{female}(y) \wedge y = x]$$

8.7 L$_\partial$: 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$_{\text{Pred}}$ and L$_\lambda$, the semantic values of expressions in L$_\partial$ 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 non-logical constants of the language to denotations of the appropriate kind based on this set of individuals and the set of truth values.

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. As before, 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.

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. For L_∂, 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. For atomic types, the fixed-up domains are defined as the union of the classical domains with the undefined entity of the corresponding type:

- $D^+_t = (D_t \cup \{\#_t\})$.
- $D^+_e = (D_t \cup \{\#_e\})$.

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. 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.

A MODEL for L_∂ based on D, then, is a triple

$$\langle (D_\tau)_{\tau\in\mathcal{T}}, \langle (D_\tau)^+_{\tau\in\mathcal{T}}, I\rangle$$

where:[6]

- $(D_\tau)_{\tau\in\mathcal{T}}$ 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.

Thus every non-logical constant of type τ has a denotation in the fixed-up domain D^+_τ. In particular, expressions of type t will have a denotation in $D_t \cup \{\#\}$. Similarly, expressions of type e will have a denotation in D^+_e, which is $D_e \cup \{\#_e\}$. Even though expressions of type τ generally may have a denotation anywhere in the fixed-up domain D^+_τ, the classical domains play a role in the theory as well, for example in the rule for quantification (see definition below).

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. Now, 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

[6]This formalization is inspired by Gallin (1975), p. 12.

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.

That said, there are many odd functions that we might not want to sanction as possible values for our non-logical constants like smokes and loves. For example, imagine a predicate of type $\langle e,t\rangle$ that maps both John and the undefined individual to T, Mary to F, and Bill to $\#$. This seems like a rather absurd denotation for a noun or a verb. To exclude such denotations, we assume that constants denoting functions must denote functions that are TOTAL in LaPierre's (1992) sense: A function f from D^+_σ to D^+_τ is TOTAL if and only if $f(k)$ is not $\#_\tau$ for all k in the classical domain D_σ. We also require for non-logical constants that their denotations are STRICT functions; A function f from D^+_σ to D^+_τ is STRICT if and only if $f(\#_\sigma)$ is $\#_\tau$.

Now onto assignments. An assignment g is a total function whose domain consists of the variables of the language such that if u is a variable of type τ then $g(u) \in D^+_\tau$.[7]

The semantic rules are the following.

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**
   $[\![\iota u.\phi]\!]^{M,g}$

[7] It is important that assignment functions are able to map variables to the undefined entity in order to ensure that beta-reduction is valid. For example, 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.

$$= \begin{cases} d & \text{if } [\![\phi]\!]^{M,g[u \mapsto d]} = \mathsf{T} \text{ but} \\ & \text{for all } d' \in D_\tau \text{ distinct from } d, [\![\phi]\!]^{M,g[u \mapsto d']} = \mathsf{F} \\ \\ \#_e & \text{otherwise} \end{cases}$$

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 8.1.

   (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 8.4.

   (c) Semantics for the unary connectives ∂ and $\mathbf{A}$ are defined as in Table 8.3.

6. **Quantification**

   (a) If u is a variable of type τ, and ϕ a formula:

   $$[\![\forall u \,.\, \phi]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\phi]\!]^{M,g[u \rightarrow k]} = \mathsf{T} \text{ for all } k \in D_\tau \\ \# & \text{if } [\![\phi]\!]^{M,g[u \rightarrow k]} = \# \text{ for some } k \in D_\tau \\ \mathsf{F} & \text{otherwise} \end{cases}$$

(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.

Exercise 16. Define a semantics for $\to$ and $\leftrightarrow$ based on the Strong Kleene connectives. Make sure that $[\phi \to \psi]$ is equivalent to $[\neg\phi \vee \psi]$ and $[\phi \leftrightarrow \psi]$ is equivalent to $[[\phi \to \psi] \wedge [\psi \to \phi]]$.

Exercise 17. Define a semantics for the universal and existential quantifiers based on the Strong Kleene connectives. Make sure that the two quantifiers are duals of each other, so $\forall x \neg\phi$ is equivalent to $\neg\exists x\phi$ and $\neg\forall x\phi$ is equivalent to $\exists x\neg\phi$. (You don't need to prove that they are duals in your answer.)

$\to$	T	F	#
T	T	F	#
F	T	T	#
#	#	#	#

$\leftrightarrow$	T	F	#
T	T	F	#
F	F	T	#
#	#	#	#

Table 8.4: Truth tables for $\to$ and $\leftrightarrow$ in Weak Kleene logic

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) $[\![\iota x\,.\,\text{elevator}(x)]\!]^{M_1}$

(b) $[\![\exists x\,.\,\text{elevator}(x)]\!]^{M_1}$

(c) $[\![\partial(\exists x\,.\,\text{elevator}(x))]\!]^{M_1}$

(d) $[\![\exists y\,.\,\mathbf{A}(y = \iota x\,.\,\text{elevator}(x))]\!]^{M_1}$

(e) $[\![\iota x\,.\,\text{elevator}(x)]\!]^{M_2}$

(f) $[\![\exists x\,.\,\text{elevator}(x)]\!]^{M_2}$

(g) $[\![\partial(\exists x\,.\,\text{elevator}(x))]\!]^{M_2}$

(h) $[\![\exists y\,.\,\mathbf{A}(y = \iota x\,.\,\text{elevator}(x))]\!]^{M_2}$

(i) $[\![\iota x\,.\,\text{elevator}(x)]\!]^{M_3}$

(j) $[\![\exists x\,.\,\text{elevator}(x)]\!]^{M_3}$

(k) $[\![\partial(\exists x\,.\,\text{elevator}(x))]\!]^{M_3}$

(l) $[\![\exists y\,.\,\mathbf{A}(y = \iota x\,.\,\text{elevator}(x))]\!]^{M_3}$

8.8 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:

(54) 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 (54a) is undefined, (54b) 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:

(55) If there is a king of France, then the king of France is wise.

(56) 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:

(57) If France remains neutral, then the king of France is wise.

(58) 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 (57) and (55) is that in (57), the antecedent (*France remains neutral*) does not entail the consequent's presupposition (that there is a king of France). Similarly, in (58), unlike in (56), 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 (55) and (56), 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:

(59) 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 (59a) the antecedent is *France is a constitutional monarchy with a king and a queen*, which is slightly stronger (more informative) than the consequent's presupposition—just that France has a king. Still, the sentence as a whole does not carry the presupposition that there is a king of France; it gets filtered out. So the antecedent need not be identical to the consequent's presupposition; an entailment relation suffices for the presupposition to be filtered out. Similarly, in (59b), the negation of the first disjunct (*France has recently crowned its first king in centuries*) is stronger than the second disjunct's presupposition (that France has a king). Here again, the presupposition gets filtered out; the sentence as a whole does not carry the presupposition that there is a king of France. Again we see that an entailment relation suffices for the filtering to take place.

Furthermore, the entailment relation may depend on real-world knowledge or assumptions (example adapted from Karttunen 1973a):

(60) Either Geraldine is not a devout Christian or she has stopped attending services on Sundays.

The second disjunct (*she has stopped attending services on Sundays*) presupposes that Geraldine did attend services on Sundays. The first disjunct is *Geraldine is not a devout Christian*, the negation of which is *Geraldine is a devout Christian*. Together with the assumption that all devout Christians have attended services on Sundays at some point in their life, the negation of the first disjunct entails that Geraldine attends services on Sundays. But the first disjunct does not carry that entailment on its own. The generalization should thus be revised to take real-world knowledge and assumptions into account: If the negation of the first disjunct, *together with real-world knowledge and assumptions*, entails a presupposition of the second disjunct, then that presupposition gets filtered out. The analogous modification is applicable to the filtering condition on conditional.

There are ways of handling presupposition projection within the "static" type of framework we have been developing so far. As mentioned above, one way to capture cases in which presuppositions fail to project is with the assertion operator **A**. Consider the following example:

(61) Either John didn't smoke before, or he stopped smoking.

Intuitively, this sentence does not carry the presupposition that John smoked in the past. As suggested above in conjunction with example (45a), suppose we represent *John stopped smoking* with the formula $[\partial\pi \wedge \phi]$, where π represents the proposition that John smoked in the past, and ψ represents the proposition that John currently does not smoke. Then the first disjunct in (61) could be represented as $\neg\pi$, so the whole translation we would derive for (61) would be:

(62) $[\neg\pi \vee [\partial(\pi) \wedge \phi]]$

With these truth-and-definedness conditions, it is impossible for the first disjunct to be true, because the presupposition of the second disjunct contributes a presupposition that it is false. This arguably goes against the pragmatics of disjunction: Asserting a sentence of the form 'A or B' should only be felicitous in a context where A is an open possibility. Beaver & Krahmer (2001) suggest that the semantic component can provide another translation for the sentence though, one with the **A** operator:

(63) $[\neg\pi \vee \mathbf{A}(\partial(\pi) \wedge \phi)]$

With this translation, the sentence as a whole carries no presupposition that John smoked in the past. Furthermore, it doesn't violate the pragmatics of disjunction. Hence, it is this translation that a person interpreting the sentence should choose. Beaver & Krahmer's (2001) 'Floating-A Theory' allows for **A**-operators to be inserted in the compositional derivation process, providing interpretations in which presuppositions are promoted to the status of at-issue content. When pragmatic principles rule out all of the interpretations in which the presuppositions survive, the presupposition-free ones are all that remain. We refer the reader to their paper for more details on how it works.

Another approach to the projection problem has made use of *dynamic semantics*, where the denotation of a sentence is a "context change potential": a function that can update a discourse context. We present a dynamic theory of meaning in the next chapter.

9 Dynamic semantics

(Significant revisions are planned for this chapter.)

9.1 Pronouns with indefinite antecedents

In this chapter, we motivate DYNAMIC SEMANTICS,[1] where the denotation of an utterance is something that depends on and updates the current discourse context. The phenomena we focus on are pronouns with indefinite antecedents, including the famous 'donkey sentences':

(1) If a farmer owns a donkey, then he beats it.

In dynamic semantics, an indefinite noun phrase like *a farmer* introduces a new DISCOURSE REFERENT into the context, and an anaphoric pronoun or definite description picks up on the discourse referent.

Consider the following two-sentence discourse:

(2) My neighbor found **a cat**. Then **it** ran away.

So far, we have analyzed indefinite descriptions as existential quantifiers. This was Russell's (1905) treatment.

There are good reasons to favor Russell's treatment of indefinites over one on which indefinites refer to some individual, as

[1] Heim 1982b, 1983b,a; Kamp & Reyle 1993; Groenendijk & Stokhof 1990a, 1991; Muskens 1996, among others

Heim (1982b) discusses. First, it correctly captures the fact that (3) does not imply that there is a specific dog that John and Mary are both friends with.

(3) John is friends with a dog and Mary is friends with a dog.

If we assumed that *a dog* referred to some particular dog, then (3) would imply that John and Mary have a common dog-friend. Second, Russell's analysis correctly captures the fact that (4) does not say that some particular dog did not come in, in contrast to (5), which has a proper name referring to a dog and does have that implication.

(4) It is not the case that a dog came in.

(5) It is not the case that Fido came in.

Third, Russell's analysis correctly captures the fact that (6) can be true even if it is not the case that there is some particular dog that everybody owns, while (7) does not have that implication.

(6) Every child owns a dog.

(7) Every child owns Fido.

If *a dog* referred to a particular dog then (6) should mean that every child owns that dog, as in (7).

However, there are some problems. If we analyze example (2) using Russell's very sensible analysis, we will derive the following representation (assuming that *it* carries the index 3, and that a sequence of two sentences is interpreted as the conjunction of the two sentences):

(8) $\exists x[\mathsf{Cat}(x) \wedge \mathsf{Found}(\mathsf{n}, x)] \wedge \mathsf{RanAway}(v_3)$

with v_3 an unbound variable outside the scope of the existential quantifier. (It doesn't matter which variable we choose; even if we choose x, the variable will still be unbound, because it will be outside the scope of the existential quantifier.) Assuming that QR does not move quantifiers beyond the sentence level, the scope of the existential quantifier introduced by *a cat* does not extend all the way to include the variable v_3, and there is no other variable-binder to bind it.

Exercise 1. Give LF trees and derivations for the two sentences in (2). (Feel free to treat *ran away* as a single verb.) Explain why these representations do not capture the connection between the pronoun and its intuitive antecedent.

One imaginable solution to this problem is to allow QR to move quantifiers to take scope over multiple-sentence discourses, so we could get the following representation:

(9) $\exists x[\mathsf{Cat}(x) \wedge \mathsf{Found}(\mathsf{s}, x) \wedge \mathsf{RanAway}(x)]$

Regarding this imaginable solution, Heim (1982a, 13) writes the following:

> This analysis was proposed by Geach [1962, 126ff]. It implies as a general moral that the proper unit for the semantic interpretation of natural language is not the individual sentence, but the text. [The formula] provides the truth condition for the bisentential text as a whole, but it fails to specify, and apparently even precludes specifying, a truth condition for the [first] sentence.’

Heim (1982a) also presents a number of empirical arguments against this kind of treatment. One comes from dialogues like the following:

(10) a. A man fell over the edge.
b. He didn’t fall; he jumped.

What would a Geachian analysis be for a case like (10)? If we let the existential quantifier take scope over the entire discourse, we would get the denotation 'there exists an x such that x is a man and x fell over the edge and x didn't fall over the edge and x jumped'. This is self-contradictory.

Another argument that Heim makes against the Geachian analysis is based on examples like the following:

(11) a. John owns some sheep. Harry vaccinated them.
b. Susan found exactly one cat. Then it ran away.

Example (11a) is only true if Harry vaccinated all of the sheep John owns. For example, it should be false in a situation where John owns six sheep, of which Harry vaccinated three. On the Geachian analysis, the interpretation would be something along the lines, 'there exists an x such that x is a bunch of sheep and John owns x and Harry vaccinated x', which would be true in such a situation. But the English sentence would not be. The reason we don't want it to be true is that maybe John owns $x+4$ sheep, but Harry only vaccinated x; that is, the x for John doesn't necessarily mean all of his sheep. so this is not a welcome prediction. Similarly, example (11b) should be false in a situation where Susan found exactly two cats, of which exactly one ran away. But the Geachian analysis predicts it to be equivalent to *There is exactly one cat that Susan found and that ran away.*

Third, Geach's proposal would mean that existential quantifiers have different scope properties from other quantifiers. Consider the following examples:

(12) A dog came in. It lay down under the table.

(13) Every dog came in. #It lay down under the table.

(14) No dog came in. #It lay down under the table.

In neither (13) nor (14) can *it* be bound by the quantifier in the first sentence.[2] Heim (1992, 17) concludes:

[2] There is a phenomenon called *telescoping*, counterexemplifying the generalization that *every* cannot take scope beyond the sentence boundary. Examples

> The generalization behind this fact is that an unembedded sentence is always a "scope-island," i.e. a unit such that no quantifier inside it can take scope beyond it. This generalization (which is just a special case of the structural restrictions on quantifier-scope and pronoun-binding that have been studied in the linguistic literature) is only true as long as the putative cases of pronouns bound by existential quantifiers under Geach's analysis are left out of consideration.

Thus it seems that Geach's solution will not do, and we need another alternative.

So-called 'donkey anaphora' is another type of case involving pronouns with indefinite antecedents that motivates dynamic semantics. The classic 'donkey sentence' is:

(15) If a farmer owns a donkey, then he beats it.

This example is naturally interpreted as a universal statement, representable as follows:

(16) $\forall x \forall y[[\mathsf{Farmer}(x) \wedge \mathsf{Donkey}(y) \wedge \mathsf{Owns}(x, y)] \rightarrow \mathsf{Beats}(x, y)]$

But the representation that we would derive for it using the assumptions that we have built up so far would be:

(17) $[\exists x \exists y[\mathsf{Farmer}(x) \wedge \mathsf{Donkey}(y) \wedge \mathsf{Owns}(x, y)]] \rightarrow \mathsf{Beats}(x', y')$

where the existential quantifiers have scope only over the antecedent of the conditional. This analysis leaves the variables introduced

---

include:
(i) Every story pleases these children. If it is about animals, they are excited, if it is about witches, they are enchanted, and if it is about humans, they never want me to stop.
(ii) Each degree candidate walked to the stage. He took his diploma from the dean and returned to his seat.
(From Poesio & Zucchi 1992, "On Telescoping")

by the pronouns (the x' and y' in (17)) unbound; clearly it does not deliver the right denotation.

Similar problems arise with indefinite antecedents in relative clauses:

(18) Every man who owns a donkey beats it.

Exercise 2. Give a representation in L_λ capturing the intuitively correct truth conditions for (18). Then give an LF tree and a derivation for (18) using the assumptions that we have built up so far. Does this derivation give an equivalent result? If so, explain. If not, give a situation (including a particular assignment function) where one would be true but the other would be false.

According to Geach (1962), we must simply stipulate that indefinites are interpretable as universal quantifiers that can have extra-wide scope when they are in conditionals or in a relative clause. But this is more of a description of the facts than an explanation for what is happening. Moreover, it is not as if just any relative clause allows for a wide-scope universal reading of an indefinite within it:

(19) A friend of mine who owns a donkey beats it.

There is no wide-scope universal reading for *a donkey* here.

Heim's (1982b) idea is that indefinites have no quantificational force of their own, but are open formulas containing variables, which may get bound by whatever quantifier there is to bind them. This is supported by the fact that their quantificational force seems quite adaptable; witness the following equivalences:

(20) In most cases, if a table has lasted for 50 years, it will last for 50 more.
$\Longleftrightarrow$ Most tables that have lasted for 50 years will last for another 50.

(21) Sometimes, if a cat falls from the fifth floor, it survives.
$\Longleftrightarrow$ Some cats that fall from the fifth floor survive.

(22) If a person falls from the fifth floor, he or she will very rarely survive.
$\Longleftrightarrow$ Very few people that fall from the fifth floor survive.

However, on Heim's view, indefinites are unlike pronouns in that they introduce a 'new' referent, while pronouns pick up an 'old' referent. This idea of novelty is formulated in the context of dynamic semantics, where as a sentence or text unfolds, we construct a representation of the text using discourse referents. A pronoun picks out an established discourse referent. An indefinite contributes a new referent, and has no quantificational force of its own. The quantificational force arises from the indefinite's environment.

The idea of a DISCOURSE REFERENT is laid out by Karttunen (1976), which opens as follows:

> Consider a device designed to read a text in some natural language, interpret it, and store the content in some manner, say, for the purpose of being able to answer questions about it. To accomplish this task, the machine will have to fulfill at least the following basic requirement. It has to be able to build a file that consists of records of all the individuals, that is, events, objects, etc., mentioned in the text and, for each individual, record whatever is said about it.

Karttunen characterizes discourse referents as follows: "the appearance of an indefinite noun phrase establishes a *discourse referent* just in case it justifies the occurrence of a coreferential pronoun or a definite noun phrase later in the text."[3] Thus a dis-

[3]Here, Karttunen is using "coreference" in a looser manner than the one Heim & Kratzer (1998) advocate when they say that "coreference implies reference". For Karttunen, any kind of anaphor-antecedent relationship qualifies as coreference, even if reference does not take place.

course referent need not correspond to any actual individual; in this sense, a discourse referent does not necessarily imply a referent. There are examples in which the occurrence of a coreferential pronoun or definite noun phrase is justified, but no particular individual is talked about, as in *No man wants **his** reputation dragged through the mud.* A discourse referent is more like a placeholder for an individual, very much like a variable. According to Karttunen, one of the virtues of this notion is that it "allows the study of coreference to proceed independently of any general theory of extralinguistic reference" (p. 367).

Karttunen (1976) also pointed out that discourse referents have a certain LIFESPAN; they do not license subsequent anaphora in perpetuity. Here is an example where a discourse referent dies:

(23) Susan didn't find a cat and keep it. #It is black.

The pronoun *it* in the second sentence cannot refer back to the discourse referent that the *it* in the first sentence picks up. The lifespan of that discourse referent ends with the scope of negation. You might be tempted to account for this fact by assuming that anaphora can only be used if the discourse referent it links back to is assigned to a particular individual or entity. But this is not a general requirement:

(24) Susan found a cat and kept it. It is black. Susan found another cat and let it run away. It was grey.

Examples (13) and (14) above provide further cases in which one can see evidence of lifetime limitations for discourse referents. So while indefinites seem to introduce discourse referents with an unusually long life span, compared to other apparently quantificational expressions, the discourse referents they introduce aren't immortal. A good theory should account for both sides of this tension.

9.2 File change semantics

Heim's (1982b) FILE CHANGE SEMANTICS conceptualizes discourse referents as file cards, very much building on Karttunen's metaphor. In file change semantics, an indefinite introduces a new file card. Subsequent anaphoric reference updates the file card. For example, consider the discourse in (25):

(25) a. A dog bit a woman.
  b. She hit him with a paddle.
  c. It broke in half.
  d. The dog ran away.

The first sentence contains two indefinites, *a dog* and *a woman*. These trigger the introduction of two new file cards; call them file card 1 and file card 2. File card 1 is associated with the property 'dog', and 'bit 2', and file card 2 is associated with the property 'woman', and 'bitten by 1'. Pictorially, we can represent the situation like this:

1
dog bit 2

2
woman bitten by 1

After the second sentence, a third card is added, and the first two cards are updated thus:

1
dog bit 2 was hit by 2 with 3

2
woman bitten by 1 hit 1 with 3

3
paddle used by 2 to hit 1

And so forth, so that by the end of the discourse, the file looks like this:

(26)

1	2	3
dog bit 2 was hit by 2 with 3 ran away	woman bitten by 1 hit 1 with 3	paddle used by 2 to hit 1 broke in half

The definite description *the dog* is assumed to behave just as an anaphoric pronoun, and the descriptive content (*dog*) serves merely to identify the appropriate discourse referent.

Exercise 3. Add a sentence to (25) and show what the file would look like afterwards.

Like Karttunen, Heim wishes to distinguish between discourse referents (i.e., file cards) and the things that they talk about. She reasons that such an identification would be absurd, because a file card is just a description and in principle it could match any number of individuals:

> what Karttunen calls "discourse referents" are, I suggest, nothing more and nothing less than file cards. Some people might disagree with this identification and maintain that discourse referents are something beyond file cards, that they are what the file cards describe. But such a distinction gains us nothing and creates puzzling questions: File cards usually describe more than one thing equally well. For example, if a card just says "is a cat" on it, then this description fits one cat as well as another.

This conception of file cards as descriptions is key to understanding how truth is conceptualized in file change semantics.

In file change semantics, it is not *formulas*, but *files* (i.e., sets of file cards), that are true or false. The truth of a file like (26) depends on whether it is possible to find a sequence of individuals that match the descriptions on the cards. For example, consider the following two worlds. Assume that in both worlds, Joan is a woman, Fido and Pug are dogs, and Paddle is a paddle.

World 1	**World 2**
Pug bit Joan	Fido bit Joan
Joan hit Pug with Paddle	Joan hit Fido with Paddle
Paddle broke in half	Paddle broke in half
Pug ran away	Fido ran away

In both worlds, it is possible to find a sequence of individuals that match the descriptions. In World 1, the sequence is ⟨Pug, Joan, Paddle⟩ (corresponding respectively to file cards 1, 2, and 3), and in World 2, it is ⟨Fido, Joan, Paddle⟩. So the file is true relative to both worlds.

More technically, we say that a given sequence of individuals SATISFIES a file in a given possible world if the first individual in the sequence fits the description on card number 1 in the file (according to what is true in the world), the second individual fits the description on card 2, etc. A file is TRUE (a.k.a. SATISFIABLE) in a possible world if and only if there is a sequence that satisfies it in that world.

On this view, the denotation of a sentence corresponds to an update to the file in the discourse. It is not any particular file; rather the denotation of a sentence constitutes a set of instructions for updating a given file. In other words, the denotation of a sentence is constituted by its potential to update the context: a CONTEXT CHANGE POTENTIAL. In file change semantics, the context is represented as a file, so the denotation of a sentence is a FILE CHANGE POTENTIAL. To make this precise, we need a conceptualization of files that is amenable to formal definitions. The boxes we have drawn give a rough idea, but they do not lend themselves to this purpose. We therefore identify a file with the *set of world-sequence pairs* such that the sequence satisfies the file in the world. For instance, the pair consisting of World 1 and the sequence ⟨Pug, Joan, Paddle⟩ would be in the set of world-sequence pairs making up the file represented by (26). So would the pair consisting of World 2 and the sequence ⟨Fido, Joan, Paddle⟩. As the denotation of a sentence in a dynamic framework is something that relates an input context to an output context, the denotation would thus be a relation between two sets of world-sequence pairs.

Recall that in a static framework, the denotation of a sentence can be identified with a set of world-assignment pairs (or model-assignment pairs): We talk about (the translation of) a sentence as being true with respect to model M and assignment function g. The set of model-assignment pairs that satisfy the formula represent the truth conditions for the sentence. Now, notice that a sequence of individuals is very much like an assignment function, mapping variables to individuals. Thus the difference between static semantics and dynamic semantics can be seen as follows: Whereas in static semantics, the denotation of a sentence corresponds to a *set of* world-assignment pairs, the denotation of a sentence in dynamic semantics corresponds to a *relation between* world-assignment pairs.

9.3 Compositional DRT

File change semantics is one theoretical framework that embodies dynamic semantics; another is DISCOURSE REPRESENTATION THEORY (Kamp & Reyle, 1993), in which DISCOURSE REPRESENTATION STRUCTURES take the place of files. Discourse representation structures (DRSs) are in a way one big file card, with information about all of the discourse referents all combined together. For example, the DRS for the discourse in (25) would look as follows:

x y z
Woman(x)
Dog(y)
Paddle(z)
Bit(y,x)
Hit-with(x,y,z)
Ran-away(y)

Just as in file change semantics, this kind of structure is thought to be built up over the course of a discourse, and the denotation of a sentence can be seen as its potential to affect any DRS representing the current state of the discourse. A DRS has two parts:

- a UNIVERSE, containing a set of discourse referents;
- a SET OF CONDITIONS, which can be simple, like Woman(x), or complex, like $\neg K$ or $K \Rightarrow K'$, where K and K' are both DRSs.

An indefinite adds a new discourse referent to the universe, and subsequent anaphora can update the information associated with that discourse referent. So, spoken out of the blue, a sentence with two indefinites like *a farmer owns a donkey* would give rise to the following DRS:

(27)

x y
Farmer(x)
Donkey(y)
Owns(x,y)

Informally, a DRS K is considered to be true in a model M if there is a way of associating individuals in the universe of M with the discourse referents of K so that each of the conditions in K is verified in M. An EMBEDDING is a function that maps discourse referents to individuals (like an assignment or sequence). The domain of this function will always be some set of discourse referents, but it may or may not include all of the possible discourse referents. In this sense, the function may be a PARTIAL FUNCTION on the set of discourse referents.

The sentence *a farmer owns a donkey* can also be used as the antecedent of a conditional, as in the famous donkey-sentence, *If a farmer owns a donkey then he beats it.* In that case it would appear as a DRS contained in a larger DRS, as follows:

(28)
$$\begin{array}{|c|}\hline \\ \hline \begin{array}{|c|}\hline \mathsf{x\ y} \\ \hline \mathsf{Farmer(x)} \\ \mathsf{Donkey(y)} \\ \mathsf{Owns(x,y)} \\ \hline \end{array} \Rightarrow \begin{array}{|c|}\hline \\ \hline \mathsf{Beats(x,y)} \\ \hline \end{array} \\ \hline \end{array}$$

This is a DRS with one condition of the form $K \Rightarrow K'$, where K and K' are themselves DRSs. Within DRT, special interpretations are stipulated for conditions of this form, and for conditions of various other forms as well.

In this section, we show that it is possible to formalize DRT within a version of Montague semantics that is based on classical type logic. One advantage of doing this is that the resulting system can be combined with other parts of the system in this book. Moreover, the formalization allows us to avoid the level of discourse representations that is specific to file change semantics and DRT, and to cut down on special-purpose auxiliary notions involved in interpreting DRT. There are many formalizations that combine DRT and Montague semantics, e.g. Dynamic Montague Grammar (Groenendijk & Stokhof, 1990b). The system we present here is based on Compositional DRT or CDRT (Muskens, 1995b, 1996). CDRT has the advantage of being based on classical logic, which makes it easy to integrate it with the system developed in the other chapters in this book. We will focus on accounting for cases of anaphora where the antecedent doesn't c-command the pronoun (as in donkey sentences) or isn't in the same sentence as the pronoun.

Formally, we will be working in a many-sorted version of the logic in Church (1940). The two-sorted version of this logic was studied in Gallin (1975); it is called two-sorted because it uses two basic types, e for individuals and t for truth values. To this we will now add a third basic type, r, which will contain discourse referents and names. Conceptually, individuals are entities of the familiar kind (like kings and cabbages) while discourse referents and names are symbols that encode the focus of our attention throughout discourse. *Discourse referents* are introduced by indefinites, while *names* are introduced by proper nouns.[4] For each type-e constant in our language (john, mary etc.), we assume that the domain also contains a type-r name r_{john}, r_{mary}, etc. In addition to names, we will assume that any model contains either an unlimited supply of discourse referents, or in any case one that is sufficient for the purpose of any discourse. To keep things simple, we will start with models that contain just three discourse referents r_1, r_2, r_3.

Alongside discourse referents, our language will still make use of variables x, y, z, x',etc., as in Chapter 4; it is important not to confuse them with discourse referents. In particular, variables can be free or bound by quantifiers and by lambda terms, but discourse referents cannot. [5]

From the three basic types e, t, r, we derive functional types as in Chapter 5. In particular, we will make use of the type $\langle r, e\rangle$, which is the type of functions from discourse referents to individuals. We will refer to objects of this type as *assignments*. These assignments are similar to the interpretation functions in Chapter

[4]Muskens (1996) uses the terms *unspecific discourse referents* for our discourse referents and *specific discourse referents* for our names.

[5]Since discourse referents and names are objects in the model and since variables can range over objects of any type, we could in principle also introduce variables of type r that range over them. Here we avoid doing so to reduce confusion and because we will not have a need for such variables.

4, with the difference that they are now considered entities within a model, alongside ordinary individuals and truth values. For reasons that will become clear shortly, we will use i, j, k, and o as variables over assignments.

We assume that every assignment maps every name to the relevant individual in the domain. By contrast, discourse referents can be mapped by different assignments to different individuals. Treating names and discourse referents in similar ways and giving them the same type allows us to let pronouns and other anaphoric expressions behave in a uniform way regardless of whether their antecedents are proper names or indefinites.

CDRT assignments are similar to the assignment functions that we introduced in Chapter 4 in that they keep track of which symbols stand for which entities. One can think of an assignment as a register that gets updated throughout the discourse. Expressions that can act as potential antecedents, such as indefinites, update the values of discourse referents in assignments, and expressions such as pronouns and definite descriptions retrieve the values of discourse referents. While each assignment is taken to be immutable (like a book that has been published and whose text cannot be edited anymore), we can simulate the process of making a change to an assignment by finding another assignment that is just like the first in all relevant respects other than the relevant change. This is encapsulated in the following definition:

(29) **Definition**
Let i and o be two assignments and r a discourse referent. We write $i[r_1]o$ to say that i and o differ at most in the value they assign to r (i.e., either i and o agree on everything except r or they do not differ at all).

There are also differences between assignments in the sense of this chapter and assignment functions as we used them in Chapter 4. Assignments in this chapter are contained in the domain of our models, just like individuals, truth values, predicates, relations, and so on. By contrast, the assignment functions that we used in Chapter 4 are not contained in our models; they are used only as devices for interpreting predicate logic formulas. Another difference is that the assignments in this chapter apply to discourse referents (of type r) while the assignment functions in Chapter 4 apply to variables of type e.

The semantics of sentences in CDRT In Section 9.2, the difference between static and dynamic semantics was summarized as follows: Whereas in static semantics, the denotation of a sentence corresponds to a *set of* world-assignment pairs, the denotation of a sentence in dynamic semantics corresponds to a *binary relation between* world-assignment pairs. Keeping the world constant for simplicity, we can say that the denotation of a sentence (its context-change potential) corresponds to a curried binary relation between assignments. Conceptually, a context-change potential is like a DRS. Formally, context-change potentials have the type $\langle re, \langle re, t\rangle\rangle$; we will abbreviate this type as **T** and we will use the letters p and q for variables that range over context-change potentials. We will call the first argument to a context-change potential the *input assignment* and its second element the *output assignment*, and we will use the letters i and o to symbolize them. By convention, we use the leftmost lambda slot for i and the second-to-leftmost one for o. CDRT extends this view to every subconstituent down to individual words, so that every lexical entry takes two assignments i and o as its arguments in addition to whatever other arguments it applies to. The grammar will be set up so that this property is passed up to larger constituents all the way up to sentences.

For example, consider again the discourse in (25), repeated here with discourse referents added. Following standard practice in dynamic semantics, discourse referents are superscripted in those places where they get introduced into the discourse, and subscripted in those places where they get picked up again. For convenience and following common practice in dynamic semantics, we have assumed that anaphoric links in sentences have already been resolved via coindexing before semantic interpretation takes place. This assumption helps us keep things simple to understand because it lets us treat pronouns as essentially denoting discourse referents; it is not crucial, and we could, instead let pronouns denote *variables* over discourse referents (Muskens, 2011).

(30)
a. A^{r_1} dog bit a^{r_2} woman.
b. She$_{r_2}$ hit him$_{r_1}$ with a^{r_3} paddle.
c. It$_{r_3}$ broke in half.
d. The$_{r_1}$ dog ran away.

The context-change potential of sentence (30a) consists in introducing two discourse referents r_1 and r_2, and updating the context such that whichever entity r_1 refers to is a dog, and whichever entity r_2 refers to is a woman that was bitten by r_1. In models where more than one dog and/or more than one woman fits the description, there will be more than one way to update the context. This suggests that the context-change potential is properly thought of as a relation between input and output contexts, rather than a function from input to output contexts. To keep things readable, from this point onwards for any assignment j and discourse referent r, we will write the lookup operation $j(r_1)$ as r_1^j.

Formally, sentence (30a) denotes the following context-change potential:

(31)
$$\lambda i \lambda o.\exists j.i[r_1]j \wedge j[r_2]o \\ \wedge \mathsf{Dog}(r_1^o) \wedge \mathsf{Woman}(r_2^o) \wedge \mathsf{Bite}(r_1^o, r_2^o)$$

In words, this is (the curried version of) the relation that holds between any two assignments i and o just in case they differ at most in what they assign to r_1 and r_2, and furthermore o maps r_1 to some dog and r_2 to some woman whom that dog bit.

In a model where indeed a dog bit a woman, this relation will be nonempty. To take an example at random, in a model that corresponds to World 1 in Section 9.2, in which Pug bit Joan, the following pair of assignments i_1 and o_1 will stand in the relation (31):

$$i_1 = \begin{bmatrix} r_1 & \rightarrow & \textsf{Bill} \\ r_2 & \rightarrow & \textsf{Bill} \\ r_3 & \rightarrow & \textsf{Mary} \end{bmatrix} \quad o_1 = \begin{bmatrix} r_1 & \rightarrow & \textsf{Pug} \\ r_2 & \rightarrow & \textsf{Joan} \\ r_3 & \rightarrow & \textsf{Mary} \end{bmatrix}$$

The values that i_1 assigns to r_1 and r_2 are irrelevant, and so is the value that both i_1 and o_1 assign to r_3. These values have been filled in only for concreteness. Many other assignments than i_1 and o_1 stand in the relation denoted by (31). For example, since the values that the input assignment assigns to r_1 and r_2 are irrelevant, i_1 and o_1 could be replaced by any other pair of assignments, so long as they map r_3 to the same value as each other and the second assignment still maps r_1 and r_2 to the same values as o_1 does. This means that the relation (31) will relate any input assignment to at least one output assignment. We will say that a relation that relates i to some output assignment *succeeds on i* (otherwise it *fails on i*); thus, the relation (31) succeeds on every input assignment.

The next sentence, (30b), denotes the following context-change potential:

(32) $\lambda i \lambda o . i[r_3]o \wedge \textsf{Hit-with}(r_2^o, r_1^o, r_3^o) \wedge \textsf{Paddle}(r_3^o)$

This relation holds between assignments i and o just in case they differ at most in what they assign to r_3, and furthermore o maps r_3 to some paddle which was used by whatever o assigns to r_2 in order to hit whatever o assigns to r_1.

What kinds of assignments stand in this relation? Since o and i must agree in everything except possibly r_3, they must both assign the same value to r_1, and likewise for r_2. As for r_3, it does not matter what i assigns it to, but o must assign it to the right kind of paddle.

For example, consider again a model that is like World 1, where

Joan hit Pug with Paddle. Suppose that no other hittings took place. In this model, for two assignments i_2 and o_2 to stand in the relation (32), i_2 must be exactly as below except that r_3 could also be mapped to any other value than Mary; and o_2 must be exactly as given below.

$$i_2 = \begin{bmatrix} r_1 & \to & \mathsf{Pug} \\ r_2 & \to & \mathsf{Joan} \\ r_3 & \to & \mathsf{Mary} \end{bmatrix} \quad o_2 = \begin{bmatrix} r_1 & \to & \mathsf{Pug} \\ r_2 & \to & \mathsf{Joan} \\ r_3 & \to & \mathsf{Paddle} \end{bmatrix}$$

Because of the constraints it imposes, the relation (32) does not succeed on every input assignment. In general, CDRT uses such relations as denotations of sentences that contain unresolved anaphoric dependencies (e.g. unbound pronouns such as *She*$_{r_2}$ and *him*$_{r_1}$ in (30b)).

Typically, the previous discourse will supply input assignments on which such sentences succeed. For example, the pronouns in (30b) have their antecedents in the previous sentence (30a).

To connect pronouns with their antecedents, we now combine the two denotations (31) and (32) by an operator called *sequencing* and written as a semicolon (;). This operator is introduced here as a shorthand:

(33) $\quad ; =_{def} \lambda p \lambda q \lambda i \lambda o . \exists j . p(i)(j) \wedge q(j)(o)$

This operator, which is present in many programming languages, takes two context-change potentials p and q and combines them to a new one which asserts that some assignment j can serve as both the output of p and the input of q. Mathematically, this amounts to composing the relations p and q; in procedural terms, this amounts to letting the output assignments of p serve as the input assignments of q. For example, the output assignment o_1 above is the same as the input assignment i_2; therefore, i_1 and o_2 will stand in the relation denoted by sequencing (31) with (32). That relation is the following:

(34) $\lambda i \lambda o \exists j \exists k . i[r_1]j \wedge j[r_2]k$
$\wedge \mathsf{Dog}(r_1^k) \wedge \mathsf{Woman}(r_2^k) \wedge \mathsf{Bite}(r_1^k, r_2^k)$
$\wedge k[r_3]o \wedge \mathsf{Hit\text{-}with}(r_2^o, r_1^o, r_3^o) \wedge \mathsf{Paddle}(r_3^o)$

In prose and simplifying a bit, this relation holds between two assignments i and o just in case o is the result of making minimal changes to i such that r_1, r_2, and r_3 are mapped to a dog, a woman that it bit, and a paddle that she hit it with.

Bridging principles Context-change potentials are relations between input assignments and output assignments. But we are used to thinking of sentences as simply being true or false. To know whether a given sentence is true or false in a model, we can convert its context-change potential into a truth value via the following bridging principles. The first bridging principle defines truth and falsity relative to an assignment:

(35) **Bridging Principle 1**
Let i be an assignment and ϕ be a term of type $\mathbf{T}$ (i.e. a context-change potential). ϕ is true relative to i iff there is an assignment o such that $i[\phi]o$ is true; otherwise ϕ is false relative to i.

The idea behind this principle is that if we only care whether a sentence is true given its input assignment, and not about whether it provides potential antecedents to subsequent sentences, then it does not matter what output assignments it produces.

For sentences without unresolved anaphoric dependencies, i.e. sentences without pronouns or definite descriptions in them, we can also define truth and falsity simpliciter by universally quantifying over input assignments:

(36) **Bridging Principle 2**
Let ϕ be a term of type $\mathbf{T}$ without unresolved anaphoric dependencies. ϕ is true iff it is true relative to every input assignment (in the sense of Bridging Principle 1); other-

wise it is false.

The idea here is that if a sentence is true in the intuitive sense, then we expect it to remain true no matter what input assignment we present it with.

In combination, the upshot of these two principles is that a context-change potential without unresolved anaphoric dependencies is true just in case it maps every input assignment to some output assignment. For example, according to these principles, the context-change potential in (34) is true just in case for every assignment i there is an assignment o that is just like i except that it maps r_1, r_2, and r_3 to a dog, a woman that it bit, and a paddle that she hit it with. Now suppose that indeed there exist a dog, a woman, and a paddle such that the dog bit the woman and the woman hit the dog with the paddle. Then for any input assignment i such an output assignment o can be obtained by changing i as needed so that it maps r_1 to the dog, r_2 to the woman, and r_3 to the paddle.

The reason that Bridging Principle 2 is restricted to sentences that do not have unresolved anaphoric dependencies is in order to avoid collapsing the truth conditions of pronouns and corresponding universals. Without this constraint, a sentence like (37a) would have the same truth conditions as Heraclitus' famous aphorism in (37b).

(37) a. It$_{r_1}$ is in flux.
b. Everything is in flux.

This is because (37a) is true relative to any input assignment that maps r_1 to something in flux. Suppose now that everything is in flux; then, and only then, every assignment whatsoever will map r_1 to something in flux. Suppose instead that some things are in flux and others aren't; in that case, some assignments will map r_1 to something in flux, while others will not. Accordingly, (37a) will be true (in the sense of Bridging Principle 1) with respect to some assignments but not others.

There is an intuitive connection between sentences with unresolved anaphoric dependencies in CDRT and formulas with free variables in predicate logic. Both can be true with respect to some assignments and false with respect to others. More generally, the input assignments in CDRT play an analogous role to the assignment functions in predicate logic.

Lexical entries for CDRT One of the advantages of using the lambda calculus to express context-change potentials is that we can now rely on it to generate them compositionally in the usual way. To do this, we equip each lexical entry with two extra slots λi and λo. For those lexical entries that do not introduce new discourse referents, we add a conjunct that requires $i = o$; otherwise almost any pair of assignments could serve as input and output and anaphoric information would be lost. For example, here are some nouns and intransitive verbs:

(38) a. *woman* $\rightsquigarrow \lambda x \lambda i \lambda o.\, i = o \wedge \mathsf{Woman}(x)$
b. *dog* $\rightsquigarrow \lambda x \lambda i \lambda o.\, i = o \wedge \mathsf{Dog}(x)$
c. *run-away* $\rightsquigarrow \lambda x \lambda i \lambda o.\, i = o \wedge \mathsf{Run\text{-}away}(x)$

The type of these entries is $\langle e, \mathbf{T}\rangle$ (recall that we use $\mathbf{T}$ to abbreviate $\langle\langle r, e\rangle, \langle\langle r, e\rangle, t\rangle\rangle$, the type of context-change potentials).

Proper nouns simply denote the relevant individuals, as usual:

(39) *John* $\rightsquigarrow \mathsf{john}$

Indefinites introduce discourse referents r by operating on the input assignment i and by using an intermediate assignment j that is constrained to differ from i at most in r. They also take a restrictor R and a nuclear scope N, both of type $\langle e, \mathbf{T}\rangle$, pass the value of r according to j to R and N and link them up via sequencing.

(40) $a^{r_1} \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j.\, i[r_1]j \wedge (R(j_r); N(j_r))(j)(o)$

For the sake of readability, from here on we will write $\phi(i)(o)$ as $i[\phi]o$, for any formula ϕ of type **T**; thus this above simplifies as follows:

(41) $a^{r_1} \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j . i[r_1]j \wedge j[R(j_r); N(j_r)]o$

We can also spell out the sequencing shorthand to make things clearer:

(42) $a^{r_1} \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j .$
$i[r_1]j \wedge \exists k . j[R(j_r)]k \wedge k[N(j_r)]o$

Using this entry and two instances of function application, sentence (43a) evaluates to (43b), which is equivalent to (43c) due to the equivalences between assignments:

(43) a. A^{r_1} dog ran away.
b. $\lambda i \lambda o . \exists k . i[r_1]j \wedge \exists k . \mathsf{Dog}(j_r) \wedge j = k$
$\wedge \mathsf{Run\text{-}away}(j_r) \wedge k = o$
c. $\lambda i \lambda o . i[r_1]o \wedge \mathsf{Dog}(r_1^o) \wedge \mathsf{Run\text{-}away}(r_1^o)$

That (43b) is so much more complicated than (43c) is due to the fact that neither the restrictor nor the nuclear scope of the indefinite *a* in (43a) happen to contain any indefinites or anything else that introduces discourse referents. In general, though, this is not always the case; and this is also the reason for the sequencing operator in (41). The point of sequencing *R* and *N* is to preserve any anaphoric links from within *R* into *N*, such as the link between *a donkey* and *it* in examples like the following:

(44) A^{r_1} [$_{Restr}$ farmer who had a^{r_2} donkey] [$_{Nucl}$ beat it$_{r_2}$].

Before we get to such examples, we will build up the rest of our lexicon as we need it for our toy discourse. Consider first pronouns. We will ignore gender and case features and simply treat them as devices that query an input assignment for the value of the discourse referent they are indexed with. We could let the pronoun just return this value, but this would prevent them from combining with predicates such as verb phrases; such predicates expect an individual, not a relation between assignments and individuals. To remedy this, we let the pronoun take its predicate as an additional argument. (This is called Montague-lifting the pronoun; we will discuss it in more detail in Chapter 10 under the heading *entity-to-quantifier shift*.). Here, P is of type $\langle e, \mathbf{T}\rangle$; thus, the type of any pronoun is $\langle\langle e, \mathbf{T}\rangle, \mathbf{T}\rangle$. In general, all noun phrases in CDRT are of this type.

(45) *he*$_{r_1}$/*him*$_{r_1}$/*she*$_{r_1}$/*her*$_{r_1}$/*it*$_{r_1}$ $\rightsquigarrow \lambda P \lambda i \lambda o \,.\, i = o \wedge i[P(i_r)]o$

For example, (46a) denotes (46b):

(46) a. It$_{r_1}$ ran away.
 b. $\lambda i \lambda o \,.\, i = o \wedge \textsf{Run-away}(i_r)$

Pronouns can also be indexed with names rather than discourse referents. Recall that our model contains names like $r^{\textsf{john}}$ that every assignment maps to the relevant individual, so that for any assignment i we have $i_{r^{\textsf{john}}}$ =John. This means that (47a) is equivalent to (47b):

(47) a. *he*$_{r^{\textsf{john}}}$ $\rightsquigarrow \lambda P \lambda i \lambda o \,.\, i = o \wedge i[P(i_{r^{\textsf{john}}})]o$
 b. *he*$_{r^{\textsf{john}}}$ $\rightsquigarrow \lambda P \lambda i \lambda o \,.\, i = o \wedge i[P(\textsf{john})]o$

Turning now to definite descriptions, we assume following Heim (1982b) that they behave just as anaphoric pronouns do, except that they come with additional descriptive content. Formally, definite determiners combine with a restrictor and a nuclear scope, which are both applied to the entity they refer to.

(48) *the*$_{r_1}$ $\rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j \,.\, i[R(j_r)]j \wedge j[N(j_d)]o$

Consider now a transitive verb such as *bite*. Following the same reasoning as before, we arrive at the following lexical entry:

(49) Preliminary entry
bite $\rightsquigarrow \lambda y \lambda x \lambda i \lambda o\,.\, i = o \wedge \mathsf{Bite}(x, y))$

This entry cannot combine with noun phrases, since they are of type $\langle\langle e, \mathbf{T}\rangle, \mathbf{T}\rangle$ rather than e. To avoid this type mismatch, we apply type shifting to the lexical entries of transitive and ditransitive verbs (for convenience, *hit with* is treated as if it was a ditransitive verb). To do so, we use the Hendriks schema presented in Section 7.4.2 to generate an Object Raising rule that is adapted for the dynamic setting. This results in the following entry. Here, Q is of type $\langle\langle e, \mathbf{T}\rangle, \mathbf{T}\rangle$.

(50) Final entry
bite $\rightsquigarrow \lambda Q \lambda x\,.\, Q(\lambda y \lambda i \lambda o\,.\, i = o \wedge \mathsf{Bite}(x, y))$

In the same way, we can use Hendriks' schema to lift the direct and indirect objects of ditransitive verbs:

(51) Preliminary entry
hit-with $\rightsquigarrow \lambda y \lambda z \lambda x \lambda i \lambda o\,.\, i = o \wedge \mathsf{Hit\text{-}with}(x, y, z)))$

(52) Final entry
hit-with $\rightsquigarrow \lambda Q' \lambda Q \lambda x\,.$
$Q'(\lambda y \lambda i \lambda o\,.\, i = o \wedge Q(\lambda z \lambda i \lambda o\,.\, i = o \wedge \mathsf{Hit\text{-}with}(x, y, z)))$

Using these entries, we can generate context change potentials for the sentences in (30). We have already seen the context-change potentials for (30a) and (30b) in (31) and (32). The one for (30c) is analogous to the one in (46b), and the one for (30d) is similar.

Exercise 4. Using the appropriate CDRT lexical entries, give a compositional derivation of the context change potential of Sentence (30d). Show the details of the derivation. Use equivalences between assignments to simplify the result as much as possible in the same manner shown in (43b) and (43c).

Exercise 5. We have seen how the context change potentials of sentences (30a) and (30b) can be combined using sequencing, as well as some examples of assignments that can serve as inputs and outputs to each of these sentences, with the output of (30a) serving as input to (30b). Do the same for the transitions from (30b) to (30c), and from (30c) to (30d). Using repeated sequencing, produce a context-change potential for the entire discourse. Paraphrase its truth conditions. Explain how anaphoric dependencies are realized and preserved.

An advantage of CDRT is that negation and conditionals do not require us to define any special composition rules. We can rely on function application for these operators just as for any other lexical entry. The following entry for *not* assumes the VP-internal subject hypothesis:

(53) $\textit{not} \rightsquigarrow \lambda p \lambda i \lambda o \,.\, i = o \wedge \neg \exists j \,.\, i[p]j$

Exercise 6. Modify this entry so that it is able to combine with a subject of type $\langle\langle e, \mathbf{T}\rangle, \mathbf{T}\rangle$ along with a VP that expects a subject of that type.

This entry limits the lifespan of discourse referents in its scope so that they are no longer available for pronouns in subsequent sentences to pick up:

(54) Paul does not own a^{r_1} donkey. #It$_{r_1}$ is grey.

Using analogous lexical entries to the ones we have already seen, we combine the transitive verb with the indefinite object and get the following:

(55) *own a*^{r_1} *donkey*
$\rightsquigarrow \lambda x \lambda i \lambda o . \exists j . i[r_1]j \wedge \mathsf{Donkey}(j_{r_1}) \wedge \mathsf{Owns}(x, j_{r_1})$

After combining with the subject, we get:

(56) *Paul owns a*^{r_1} *donkey*
$\rightsquigarrow \lambda i \lambda o . . i[r_1]o \wedge \mathsf{Donkey}(r_1^o) \wedge \mathsf{Owns}(\mathsf{paul}, r_1^o)$

This context-change potential relates any two assignments i and o just in case they differ at most in r, in such a way that o maps r to a donkey that Paul owns.

The bridging principles in (35) and (36) have the effect that this is true just in case there exist an assignment i and an assignment o that stand in this relation.

Negation now applies and converts this into a context-change potential that requires i and o to be identical, and furthermore ensures that there is no assignment that is like i aside from mapping r to a donkey Paul owns:

(57) *not(Paul owns a*^{r_1} *donkey)*
$\rightsquigarrow \lambda i \lambda o . i = o \wedge \neg \exists j . i[r_1]j \wedge \mathsf{Donkey}(j_{r_1}) \wedge \mathsf{Owns}(\mathsf{paul}, j_{r_1})$

The bridging principles have the effect that this is true just in case there is an assignment i such that for no assignment o is it the case that i differs from o at most in that o maps r to a donkey Paul owns. That is to say, there is an assignment i such that every assignment o differs from i in more than the fact that o maps r to a donkey Paul owns.

This chapter has only given a taste of dynamic semantics, enough to show that it has the power to deal smoothly with the apparently variable force of indefinites. Geurts & Beaver (2011) provide a more thorough overview, including more on the notion of 'accessiblity', which constrains the 'lifespan' of discourse referents. The interested student is encouraged to start there and work backwards from the references cited there.

10 Coordination and plurals

10.1 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 CoordP
CoordP	→	Coord S

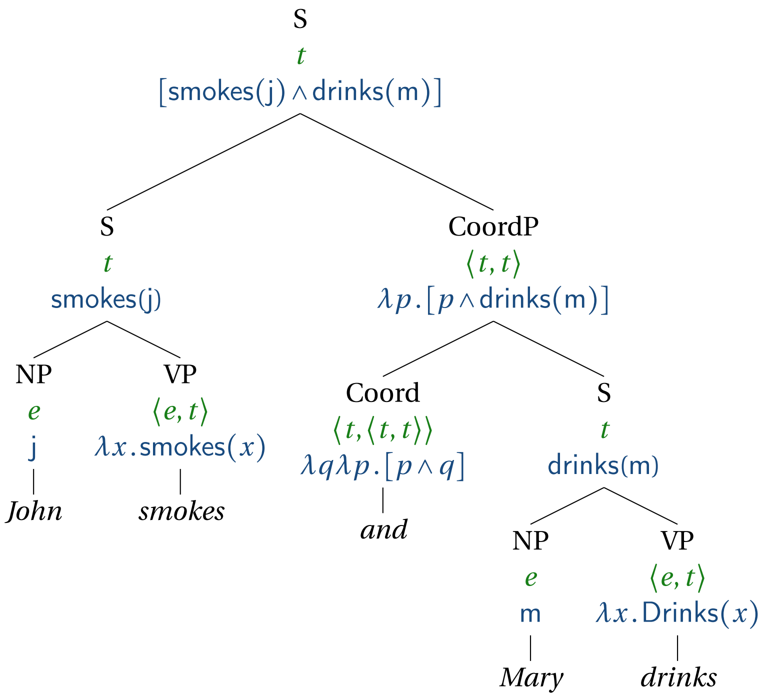
(2) **Lexicon**
Coord: *and*, *or*

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 by pairs of rules of the following kind, one pair for each category:

(9) **Syntax**

X	$\to$	X CoordP
CoordP	$\to$	Coord X

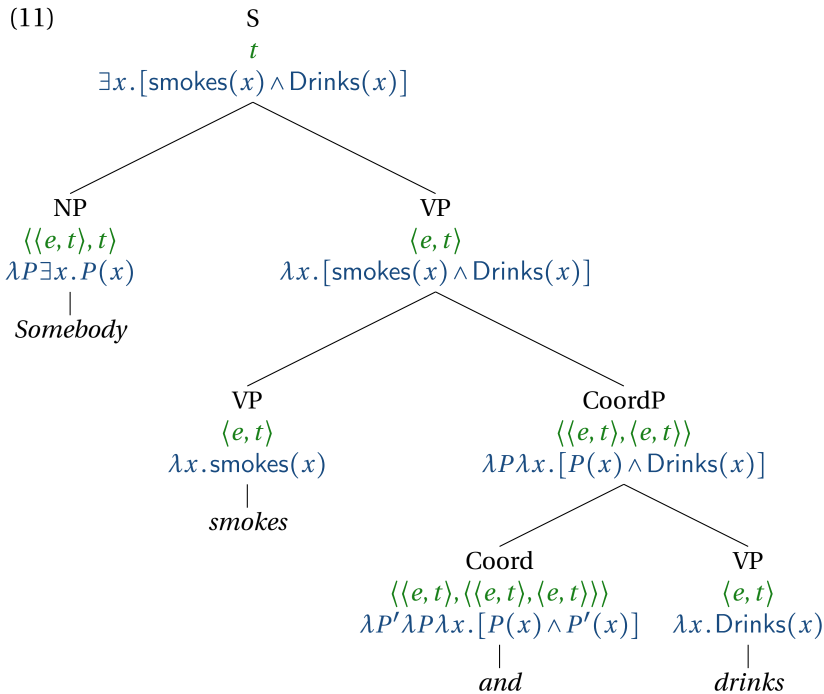
where $X \in \{S, VP, DP, V, \dots\}$

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. $\textit{and}_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . [R(y)(x) \wedge R'(y)(x)]$
b. $\textit{or}_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . [R(y)(x) \vee R'(y)(x)]$

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 conjunctions 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. $\textit{and}_{DP} \rightsquigarrow \lambda Q' \lambda Q \lambda P . [Q(P) \wedge Q'(P)]$
b. $\textit{or}_{DP} \rightsquigarrow \lambda Q' \lambda Q \lambda P . [Q(P) \vee Q'(P)]$

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 4. 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) and has made a brief appearance in Chapter 9, 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:

$$[\mathsf{smokes}(\mathsf{j}) \wedge \mathsf{smokes}(m)]$$

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.

10.2 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:

$$[\mathsf{happy\text{-}couple}(\mathsf{j}) \wedge \mathsf{happy\text{-}couple}(m)]$$

since this would entail $\mathsf{happy\text{-}couple}(\mathsf{j})$ 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 (1983a), 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 (1983b). 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 10.1 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).

In Figure 10.1, 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:

(22) **Axiom of reflexivity**
$\forall x[x \leq x]$
(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*.

(23) **Definition: Proper part**
$x < y \overset{\text{def}}{=} x \leq y \land x \neq y$
(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:

(24) **Axiom of transitivity**
$\forall x \forall y \forall z[x \leq y \land y \leq z \rightarrow x \leq z]$
(Any part of any part of a thing is itself part of that thing.)

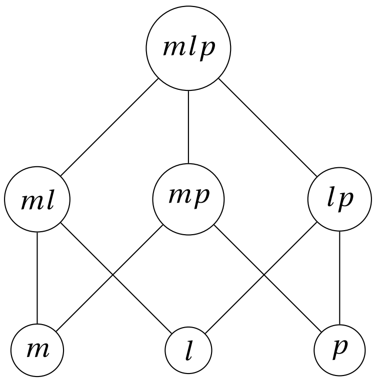
For example, according to Figure 10.1, Maxine is part of Maxine⊕LaVerne, and Maxine⊕LaVerne is part of Maxine⊕LaVerne⊕Patty; therefore, by transitivity, Maxine is also part of Maxine⊕LaVerne⊕Patty.

Finally, antisymmetry means that two distinct things cannot both be part of each other.

(25) **Axiom of antisymmetry**
$\forall x \forall y [x \leq y \wedge y \leq x \rightarrow x = y]$
(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 10.1: 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:

(26) **Binary sum**
$x \oplus y$ = that individual k such that

- $x \leq k$,
- $y \leq k$,
- there is no $k' < k$ such that $x \leq k'$ and $y \leq k'$.

So far we have been using the symbols ⊕ and ≤ 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 ≤, which in turn determines a sum operation ⊕, 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 ≤:

$$[\![\alpha \leq \beta]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g} \leq [\![\beta]\!]^{M,g} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Semantic rule: Sum
If α and β are expressions of type e and M determines the sum operation ⊕:

$$[\![\alpha \oplus \beta]\!]^{M,g} = [\![\alpha]\!]^{M,g} \oplus [\![\beta]\!]^{M,g}.$$

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.*

10.3 The plural

10.3.1 Algebraic closure

Coordinations of proper nouns are not the only way to talk about sums:

(27) 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 (27a) fully specifies the parts of that sum, while (27b) through (27d) 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:

(28) **Definition: Generalized sum**
Relative to a given model M determining the part relation $\leq$, given a nonempty set $S \subseteq D_e$,

$\bigoplus S \overset{\text{def}}{=}$ the individual k in D_e such that:
(i) for all $x \in S$, $x \leq k$;
(ii) there is no $k' < k$ such that for all $x \in S$, $x \leq k'$.

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. For doing so, it is convenient to have a way of notating the characteristic set of a function. Let $\mathsf{Set}(P)$ denote the characteristic set of P, so $\mathsf{Set}(P) = \{x \mid P(x) = \mathsf{T}\}$.

Semantic rule: Generalized sum
If π is an expression of type $\langle e, t\rangle$, then:
$[\![\bigoplus \pi]\!]^{M,g} = \bigoplus \mathsf{Set}([\![\pi]\!]^{M,g})$

Every expression of the form $\bigoplus \pi$ 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'.

(29) **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. We use $\mathsf{Fun}(S)$ to denote the characteristic function of a set S (the function f such that $f(x) = \mathsf{T}$ just in case $x \in S$).

Semantic rule: Star operator
If π is an expression of type $\langle e,t\rangle$, then:
$[\![{}^*\pi]\!]^{M,g} = \mathsf{Fun}({}^*\mathsf{Set}([\![\pi]\!]^{M,g}))$

In other words, if S is the characteristic set of $[\![\pi]\!]^{M,g}$, then $[\![{}^*\pi]\!]^{M,g}$ is the characteristic function of $^*S$. 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:

(30) **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:

$$\{m, l, p, m \oplus l, m \oplus p, l \oplus p, m \oplus l \oplus p\}$$

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 we will continue to use (30) as the denotation of the plural, and 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:

(31) 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 (31a), 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:

(32) a. Maxine is a singer and LaVerne is a singer.
$\rightsquigarrow$ $[\textsf{singer}(\textsf{mx}) \wedge \textsf{singer}(\textsf{l})]$
b. Maxine and LaVerne are singers.
$\rightsquigarrow$ ${}^{*}\textsf{singer}(\textsf{mx} \oplus \textsf{l})$

Exercise 8. Draw trees for the sentences in (31) and (32b), 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 (31) and (32b).

10.3.2 Plural definite descriptions

Now, supposing that *singers* denotes a predicate that holds of any singer-plurality, what does *the singers* denote? If *the singers* is translated as:

$$\iota x\,.\,{}^{*}\textsf{singer}(x)$$

then a presupposition failure will arise as long as there is more than one singer, because more than one individual will satisfy the predicate ${}^{*}\textsf{singer}$.[1] 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:

(33) $\text{the}_{\text{SUM}} \rightsquigarrow \lambda P\,.\bigoplus P$

[1]This was pointed out by Sharvy (1980).

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:

(34) the_{SUM} singers $\rightsquigarrow \bigoplus {}^{*}\mathsf{singer}(x)$

In a model where the singers are Maxine, LaVerne and Patty, (34) denotes $\{m, l, p, m \oplus l, m \oplus p, l \oplus p, m \oplus l \oplus p\}$. As can be checked with Figure 10.1, the sum of this set, and therefore the denotation of *the singers*, 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. To see this, let us first establish some assumptions about how phrases like *two singers* are interpreted. Suppose that *two* denotes the property of being a sum of exactly two atomic individuals (for which we will write $\mathsf{card}(x) = 2$), and that it combines with *singers* via Predicate Modification:

(35) two $\rightsquigarrow \lambda x.\mathsf{card}(x) = 2$

Then *two singers* will translate as follows:

(36) 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 9. In order to deal with sentences like *Two singers met*, we can assume that there is a silent determiner with the semantics of a generalized existential quantifier:

$$\varnothing_D \rightsquigarrow \lambda P \lambda P'.\exists x.[P(x) \wedge P'(x)]$$

Give a derivation for *Two singers met* using this assumption. Don't forget to include the silent determiner in the tree diagram.

Now, what about *the two singers*? If we use *the*$_{\text{SUM}}$ from above, then we will get the sum of the two-singer pluralities. As a glance at Figure 10.1 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 10. Translate *The*$_{\text{SUM}}$ *singers met* and *The*$_{\text{SUM}}$ *two singers met.*

Exercise 11. In the model where the singers are Maxine, LaVerne, and Patty, what (if anything) do the expressions *the*$_{\text{SUM}}$ *singer*, *the*$_{\text{SUM}}$ *singers*, *the*$_{\text{SUM}}$ *two singers* and *the*$_{\text{SUM}}$ *three singers* denote? In each case, explain which presupposition arises and whether it is satisfied.

Which of these cases does this theory of plural *the* make the correct predictions for?

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:[2]

[2]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 1973, 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.

(37) $\text{the}_{\textsc{supr}} \leadsto \lambda P \iota x[P(x) \wedge \forall y[P(y) \rightarrow y \leq x]]$

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 10.1, 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 (37) picks out the only member of that singleton. In all other models, the ι operator will not be defined.

Exercise 12. Translate *The*$_{\textsc{supr}}$ *singers met* and *The*$_{\textsc{supr}}$ *two singers met.* This exercise can be solved in the Lambda Calculator.

Exercise 13. In the model where the singers are Maxine, LaVerne, and Patty, what (if anything) do the expressions *the*$_{\textsc{supr}}$ *singer*, *the*$_{\textsc{supr}}$ *singers*, *the*$_{\textsc{supr}}$ *two singers* and *the*$_{\textsc{supr}}$ *three singers* denote? In each case, explain which presupposition arises and whether it is satisfied.

Which of these cases does this theory of plural *the* make the correct predictions for?

10.4 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:

(38) 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 (38a) 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 (38b) (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 (38c) 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*.

Just like distributive readings, cumulative readings 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.

(39) **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.

(40) **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 $a, b \in D_e$,
$R(a)(b) = \mathsf{T}$ iff $\langle a, b\rangle \in {}^*\{\langle x, y\rangle \mid [\![\pi]\!]^{M,g}(x)(y) = \mathsf{T}\}$

We can then represent cumulative readings by using the algebraic closure of transitive verbs:

(41) Maxine, LaVerne and Patty own four toothbrushes. $\rightsquigarrow$
$\exists x\,.\,{}^*\mathsf{toothbrush}(x) \wedge \mathsf{card}(x) = 4 \wedge {}^*\mathsf{own}(\mathsf{m} \oplus \mathsf{l} \oplus \mathsf{p}, x)$

An example model which verifies formula (41) 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 $\langle \text{Maxine}, t_1 \rangle$, $\langle \text{LaVerne}, t_2 \rangle$, $\langle \text{Patty}, t_3 \rangle$ and $\langle \text{Patty}, t_4 \rangle$. The sum of these four pairs is $\langle \text{Maxine} \oplus \text{Laverne} \oplus \text{Patty}, t_1 \oplus t_2 \oplus t_3 \oplus t_4 \rangle$. So there is indeed a value for x that makes the scope of the existential claim true, namely $t_1 \oplus t_2 \oplus t_3 \oplus t_4$.

10.5 Summary

In this chapter, we have extended the syntax and semantics of our logic in order to adapt it to the new entities and relations we have added. Our new representation language is called L_*.

10.5.1 Logic syntax

The syntax of L_* is defined as in three-valued type logic (L_∂), plus 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 $\alpha \leq \beta$ 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 $\langle e, t \rangle$.

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 $\alpha < \beta$ 'alpha is a proper part of beta' as an abbreviation for $\alpha \leq \beta \wedge \alpha \neq \beta$ 'alpha is a part of beta and alpha is distinct from beta'.

10.5.2 Logic semantics

A model for L_* is a based on D is a 4-tuple

$$\langle (D_\tau)_{\tau \in \mathcal{T}}, \langle (D_\tau)^+_{\tau \in \mathcal{T}}, I, \leq \rangle$$

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.
- $\leq$ 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 $\leq$, given a set $S \subseteq D_e$,

  $\bigoplus S \stackrel{\text{def}}{=}$ the individual k in D_e such that:
  (i) for all $x \in S$, $x \leq k$;
  (ii) there is no $k' < k$ such that for all $x \in S$, $x \leq k'$.

- **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.

- **Characteristic set**
  $\mathsf{Set}(P) = \{x \mid P(x) = \mathsf{T}\}$.

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 $\leq$:

  $$[\![\alpha \leq \beta]\!]^{M,g} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g} \leq [\![\beta]\!]^{M,g} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

- **Binary sum**
  If α and β are expressions of type e and M determines the part-of relation $\leq$:

  $[\![\alpha \oplus \beta]\!]^{M,g} = [\![\alpha]\!]^{M,g} \oplus [\![\beta]\!]^{M,g}$.

- **Generalized sum**
  If π is an expression of type $\langle e,t\rangle$, then:
  $[\![\bigoplus \pi]\!]^{M,g} = \bigoplus \mathsf{Set}([\![\pi]\!]^{M,g})$

- **Star operator**
  If π is an expression of type $\langle e,t\rangle$, then:
  $[\![{}^{*}\pi]\!]^{M,g} =$ that function f such that
  for all $x \in D_e$, $f(x) = \mathsf{T}$ iff $x \in {}^{*}\{y \mid [\![\pi]\!]^{M,g}(y) = \mathsf{T}\}$

- **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 $a, b \in D_e$,
  $R(a)(b) = \text{T}$ iff $\langle a, b\rangle \in {}^*\{\langle x, y\rangle \mid [\![\pi]\!]^{M,g}(x)(y) = \text{T}\}$

10.5.3 English syntax

Syntax rules. We add the following rules for coordination:

(42) **Syntax**

X	$\rightarrow$	X CoordP
CoordP	$\rightarrow$	Coord X

where $X \in \{$S, VP, DP, V,...$\}$

In addition, we add the following rule for the plural:

$$\text{N} \quad \rightarrow \quad \text{N Pl}$$

Lexicon. Lexical items are associated with syntactic categories as follows:

D:	$\varnothing_D$
A:	*two, three* etc.
Coord:	*and, or*
Pl:	*-s*
V:	*met, own*

10.5.4 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* $\rightsquigarrow \lambda P . \iota z[P(z) \wedge \forall x[P(x) \rightarrow x \leq z]]$

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. $\varnothing_D \rightsquigarrow \lambda P \lambda Q . \exists x . [P(x) \wedge Q(x)]$

11 Event semantics

11.1 Why event semantics

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. $[\mathsf{smokes}(\mathsf{j}) \wedge \mathsf{drinks}(\mathsf{m})]$
b. $\mathsf{smokes}(\mathsf{j})$

In every model where (2a) is true, also (2b) is 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 chapters (we are treating *the toast* as if it was a constant rather than a definite description, but nothing will hinge on this):

(4) $\mathsf{butter}(\mathsf{j},\mathsf{t})$

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) $[\mathsf{butter}(\mathsf{j},\mathsf{t}) \wedge \mathsf{slow}(\mathsf{j})]$

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. $[\mathsf{butter}(\mathsf{j},\mathsf{b}) \wedge \mathsf{quick}(\mathsf{j})]$

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) $[\mathsf{slow}(\mathsf{j}) \wedge \mathsf{quick}(\mathsf{j})]$

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) $\mathsf{slow}(\mathsf{butter}(\mathsf{j},\mathsf{t}))$

In the system we have been developing so far, there is a problem with this idea too. The denotation of the subformula $\mathsf{butter}(\mathsf{j},\mathsf{t})$ 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 $\langle et, et \rangle$, and applies to the VP *buttered the toast* and then to *Jones*?

(9) $\mathsf{slow}(\lambda x.\mathsf{butter}(y, \mathsf{t}))(\mathsf{j})$

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) $\forall P \forall x.\mathsf{slow}(P)(x) \to P(x)$

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 $\langle et, et \rangle$ and accounted for their intersectivity by a meaning postulate. In that chapter, we also saw that we can give intersective adjective 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 provide 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, 1990a):

(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 treatment – 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, 1990a):

(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. $\exists e.\mathsf{violent}(e)$

11.1.1 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, 1990a). 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 Dakṣiputra 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 is being developed 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 was 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 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 intransitive, the secondis 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, 1990a):

(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 syntactic 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, 1990a). 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 10, 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, 1990a; 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.

11.2 Aktionsart

So far we have been speaking of events, but not all sentences describe events. Some describe states, as in *Ann knows French* and *Zoe is in town.* An overarching term that can describe both events and states is 'eventuality'. In a famous paper entitled *Verbs and Times,* Vendler (1957) distinguished between four types of eventualities:

- States (example: *know the answer*) are static, extended in time, and lack a natural end point.
- Activities (example: *make sandcastles*) are like states except

they typically involve or lead to some kind of change.

- Accomplishments (example: *run a mile*) are like activities except they have a natural end point.

- Achievements (example: *reach the pier*) are like accomplishments except they are punctual rather than extended in time.

A fifth type, namely ‘semelfactives’, was later added (example: *cough*). They are like achievements except they do not lead to a change. These five types of *aktionsart* are categories of states or events—EVENTUALITIES, to be neutral between state and event—with various different properties.[1]

One dimension along which these different eventuality types differ is TELICITY. A telic eventuality has a natural endpoint; *telos* means ‘goal’ in Greek. Verb phrases denoting telic eventuality types can be modified with *in*-adverbials such as *in an hour*. Compare:

(31) a. Ida ran a mile in an hour. [accomplishment]
b. ??Ida made sandcastles in an hour. [activity]

Run a mile is telic, while *make sandcastles* is not: it is *atelic*.

Verb phrases denoting *atelic* eventualities, on the other hand, are more natural in combination with *for*-adverbials such as *for an hour*:

(32) a. ??Ida ran a mile for an hour. [accomplishment]
b. Ida made sandcastles for an hour. [activity]

States, activities and semelfactives are atelic, while accomplishments and achievements are telic.

What distinguishes states from activities is that activities are DYNAMIC (they require constant influx of energy) while states are not. For example, making sandcastles or running along the beach

[1] Other words for ‘aktionsart’ include *lexical aspect, situation aspect, internal aspect, aspectual class,* and *situation type.*

requires energy, while having a friend does not. The state/non-state distinction also has reflexes in the grammar. The progressive in English does not combine well with stative predicates:

(33) a. Ida is running along the beach. [activity]
b. ??Ida is having a friend. [state]

Furthermore, the simple present tense gives rise to a habitual interpretation only with non-stative predicates:

(34) a. Ida runs along the beach. [activity: habitual]
b. Ida has a friend. [state: non-habitual]

What distinguishes accomplishments from achievements and semelfactives is that the former are DURATIVE while the latter are conceptualized as taking place essentially at a single moment. This contrast can be observed in conjunction with *in* phrases. To see this, consider the following sentences:

(35) a. Ida will run a mile in 20 minutes. [accomplishment: in = duration]
b. Ida will reach the pier in 20 minutes. [achievement: in = after]
c. Ida will jump in 20 minutes. [semelfactive: in = after]

With the accomplishment *run a mile*, 20 minutes can measure the duration of the running-a-mile event, while with the achievement *reach the pier* and the semelfactive *jump*, 20 minutes can only measure the time that will elapse before the event takes place.

Finally, what distinguishes achievements from semelfactives is that the former involve a change of state while the latter do not. Because semelfactives do not involve a change of state, they can be iterated, and an iterative reading arises with *for* adverbials:

(36) a. Ida jumped for an hour. [semelfactive: iterative]
b. ??Ida reached the pier for an hour. [achievement]

The idea of (36a) being iterative is that the sentence suggests that Ida jumped multiple times within the hour. Repeated jumping is an eventuality type that is atelic, unlike jumping once, which is telic. The repetition induced by the *for* adverbial here can be seen as a secondary operation on the denotation of the verb *jump*, taking it from its basic telic denotation to an atelic denotation involving iteration of the basic denotation.

The kind of eventuality being described can depend on the object of the verb. For example, *make sandcastles* is atelic, while *make a sandcastle* is telic. Thus it is not verbs but verb phrases that are appropriate to classify with respect to their aktionsart. But as we have just seen in the case of semelfactives, there may be other elements in a sentence that help to determine the aspectual properties of the eventuality being described by the entire sentence.

The properties of these five classes are summarized in Table 11.1, taken from Smith (1997):

	Durative	Dynamic	Telic
State	+	-	-
Activity	+	+	-
Accomplishment	+	+	+
Achievement	-	+	+
Semelfactive	-	+	-

Table 11.1: Types of eventualities

Henceforth, we may use the term ‘event’, but we generally mean ‘eventuality’.

11.3 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 adjuncts, until an existential quantifier is inserted at the end and binds the event variable (Carlson, 1984; Parsons, 1990a, 1995). A more recent approach views this existential quantifier as part of the lexical entry of the verb, and arguments and adjuncts as adding successive restrictions to this quantifier (Champollion, 2015).

Both strategies are compatible with the idea that adjuncts and prepositional phrases are essentially conjuncts that apply to the same event. We discuss both of them here. The first approach is more widespread and is sufficient for simple purposes, while the second leads to a cleaner interaction with certain other components of the grammar such as conjunction, negation and quantifiers.

There are also other strategies that we will not discuss. For example, Landman (1996) assumes that the lexical entry of a verb consists of an event predicate conjoined with one or more thematic roles. Kratzer (2000) argues that verbs denote relations between events and their internal arguments while external arguments (subjects) are related to verbs indirectly by theta roles.

11.3.1 Verbs as predicates of events

On the first strategy, verbs denote predicates of events:

(37) 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. Since these variables need to be related to the event by thematic roles, we need to provide means for these roles to enter the derivation. One way to do so is to allow each noun phrase a way to "sprout" a theta role head θ.

(38) **Syntax**
DP $\to$ θ DP

We then write lexical entries that map these heads to suitable roles:

(39) **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 map these theta roles to thematic roles:

(40) a. *[agent]* $\rightsquigarrow \lambda x \lambda e.\mathsf{agent}(e) = x$
b. *[theme]* $\rightsquigarrow \lambda x \lambda e.\mathsf{theme}(e) = x$
c. ...

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 5. Existential closure
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).

The quantifier that binds the variable e is called the EVENT QUANTIFIER. It does not correspond to anything pronounced in an English sentence.

A sample derivation that shows all of the elements we have introduced is shown in (41). 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.

(41)

S
t
$\exists e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}$
$\Uparrow$
$\langle v, t\rangle$
$\lambda e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = \mathsf{s}$

- DP, $\langle v, t\rangle$, $\lambda e.\mathsf{agent}(e) = \mathsf{s}$
  - θ, $\langle e, \langle v, t\rangle\rangle$, $\lambda x \lambda e.\mathsf{agent}(e) = x$ — [agent]
  - DP, e, s — Spot
- VP, $\langle v, t\rangle$, $\lambda e.\mathsf{bark}(e)$ — barks

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.

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:

(42) **Syntax**

S	→	AdvP S
VP	→	AdvP VP
VP	→	VP AdvP
AdvP	→	Adv

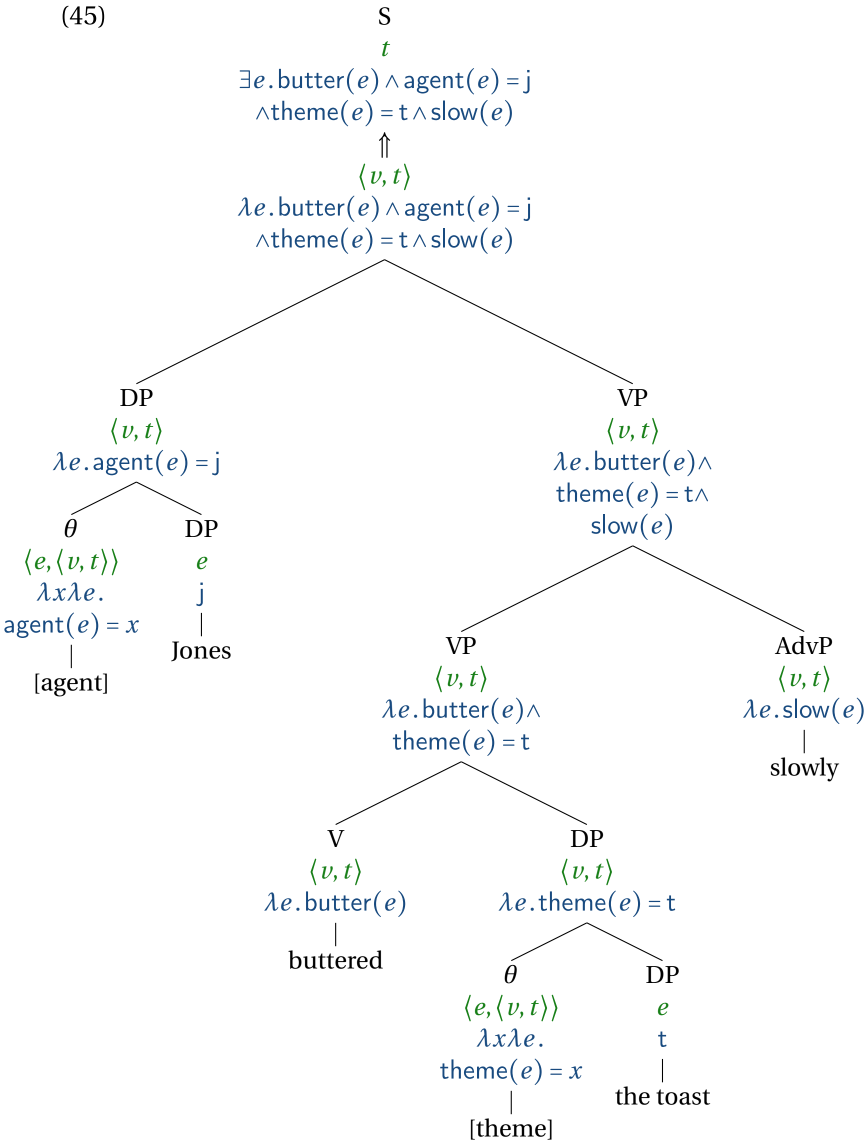
(43) **Lexicon**
Adv: slowly

As we have seen above, *slowly* is interpreted as an event predicate. Its lexical entry is therefore very simple:

(44) a. *slowly* $\rightsquigarrow \lambda e.\mathsf{slow}(e)$

The tree in (45) 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:

(45)



In the derivation in (45), 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.

11.3.2 A formal fragment

Let us recapitulate the additions to our fragment. The syntax is defined as in three-valued type logic (L_λ with three truth values as in Chapter 8), plus the following additions:

Syntax rules. We add the following rule:

(46) **Syntax**
DP $\rightarrow$ θ DP

Lexicon. Lexical items are associated with syntactic categories as follows:

θ: [agent], [theme], ...

Types. As mentioned, we add a new basic type to the system: v, the type of events. Complex types are generated from this type and the other two basic types (e and t) in the usual way. For example, $\langle v, t\rangle$ is the type of sets of events (or equivalently, functions from events to truth values); $\langle v, e\rangle$ is the type of functions from events to individuals; and so on.

Translations. Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- e is a variable of type v
- bark and butter are constants of type $\langle v, t \rangle$,
- agent, theme, and other thematic roles are constants of type $\langle v, e \rangle$. (To keep formulas readable, we depart from the practice we adopted in Chapter 4, and no longer require all function symbols to end in Of. In the literature, thematic roles are also sometimes treated as two-place predicates rather than as functions; we could have followed this approach and written $\mathsf{Theme}(e, t)$ instead of $\mathsf{theme}(e) = t$)

Type $\langle v, t \rangle$:

1. *bark* $\rightsquigarrow \lambda e.\mathsf{bark}(e)$
2. *butter* $\rightsquigarrow \lambda e.\mathsf{butter}(e)$

Type $\langle e, \langle v, t \rangle \rangle$:

1. *[agent]* $\rightsquigarrow \lambda x \lambda e.\mathsf{agent}(e) = x$
2. *[theme]* $\rightsquigarrow \lambda x \lambda e.\mathsf{theme}(e) = x$

11.4 Quantification in event semantics

The system we have seen so far is sufficient for many purposes, including the sentences discussed at the beginning of the chapter. Most papers that use event semantics assume some version of it, although the details differ. Things become more complicated, though, when we bring in quantifiers like *every cat* and *no dog*. As we have seen in Chapter 7, these quantifiers are able to take scope in various positions in the sentence. We have seen that this can be explained using quantifier raising or type-shifting. 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 (47), read with neutral intonation, is not ambiguous. Its only reading corresponds to (48b), where the event quantifier takes low scope. As for (49b), that is not a possible reading of the sentence.

(47) No dog barks.

(48) 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"

(49) 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"

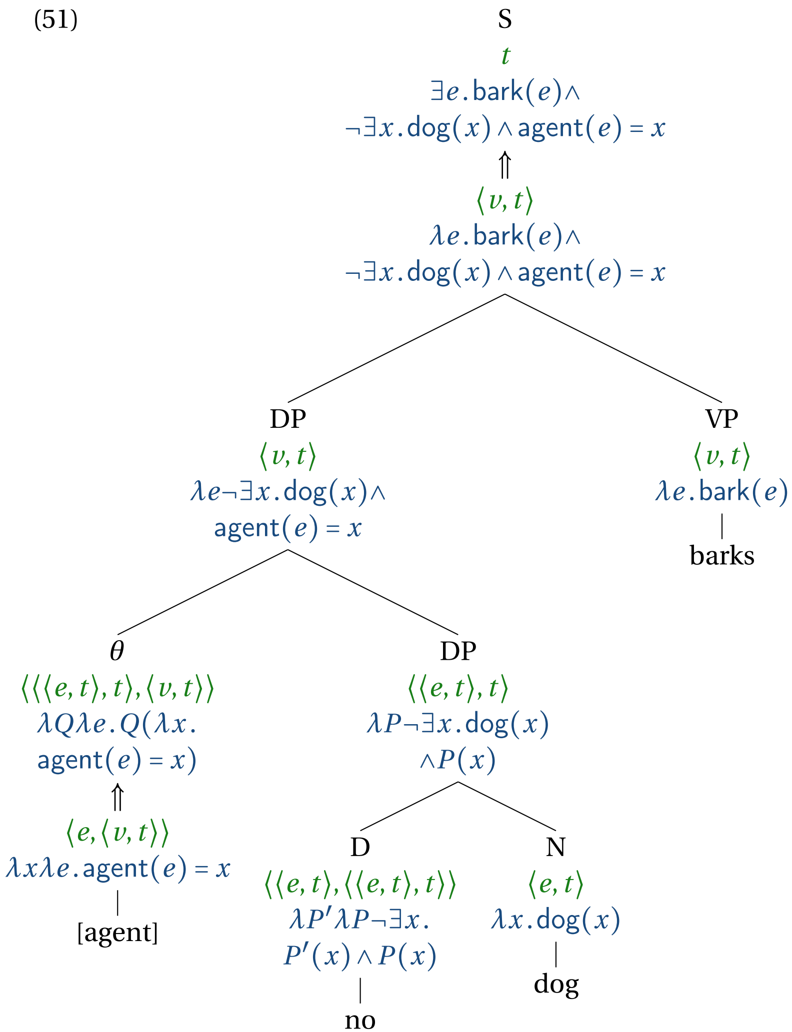
Exercise 1. How can you tell that (49b) is not a possible reading of sentence (47)?

As it turns out, each of the two strategies for the interpretation of quantifiers — quantifier raising and type-shifting — generates one of these two formulas. Quantifier raising *no dog* above the sentence level leads to the only available reading (48b), while applying Hendriks' object raising rule (or rather, the general schema) to the theta role head leads to the unavailable reading (49b). This is shown in (50) and (51), respectively.

(50)

- S, t, $\neg\exists x[\mathsf{dog}(x) \wedge \exists e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = x]$
  - DP, $\langle\langle e,t\rangle,t\rangle$, $\lambda P \neg\exists x[\mathsf{dog}(x) \wedge P(x)]$
    - D, $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$, $\lambda P' \lambda P \neg\exists x [P'(x) \wedge P(x)]$
      - no
    - N, $\langle e,t\rangle$, $\lambda x.\mathsf{dog}(x)$
      - dog
  - λP, $\langle e,t\rangle$, $\lambda v_1 \exists e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = v_1$
    - 1
    - S, t, $\exists e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = v_1$
      - $\Uparrow$ $\langle v,t\rangle$, $\lambda e.\mathsf{bark}(e) \wedge \mathsf{agent}(e) = v_1$
        - DP, $\langle v,t\rangle$, $\lambda e.\mathsf{agent}(e) = v_1$
          - θ, $\langle e,\langle v,t\rangle\rangle$, $\lambda x \lambda e.\mathsf{agent}(e) = x$
            - [agent]
          - DP, e, v_1
            - t_1
        - VP, $\langle v,t\rangle$, $\lambda e.\mathsf{bark}(e)$
          - barks

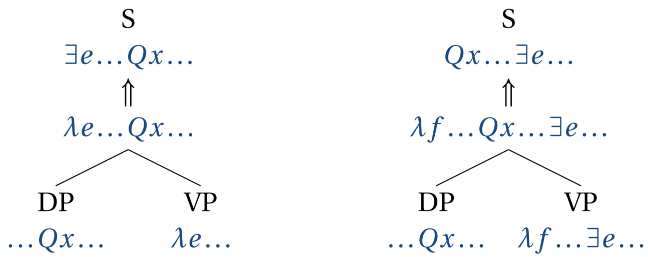
(51)



The interim conclusion, then, is that event semantics seems to commit us to a quantifier-raising based treatment of quantificational noun phrases.

11.4.1 Verbs as event quantifiers

In the tree in (50), we needed to apply quantifier raising to *no dog* in order to give it scope above the event quantifier, which was introduced by the existential-closure rule at sentence level. If the event quantifier was introduced lower than *no dog*, there would be no need to raise it. This brings us to the second strategy for the compositional treatment of event semantics, due to Champollion (2014). As mentioned, on this approach, verbs come equipped with their own event quantifiers. Verbs no longer denote event predicates but rather generalized existential quantifiers over events. Instead of sitting at the edge of the sentence, which results in the wrong relation as in Figure 11.1a, the event quantifier is now made part of the lexical entry for the verb, as in Figure 11.1b. This results in the right scope relation between quantificational noun phrases and the event quantifier and removes the need for quantifier raising. Champollion (2014) argues that this is preferable for analyses of languages in which there are no scope ambiguities, such as Chinese (Huang, 1981). For languages like English, both approaches are viable in principle.



(a) $\exists e$ introduced at S (b) $\exists e$ introduced by verb

Figure 11.1: Comparison of two approaches to event semantics. Note the position of the existential in each subfigure.

To implement this approach, we need to revise our semantics. We will equip each verb with a variable f of type $\langle v, t \rangle$, a variable over sets of events. This variable will stand 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 will include the event quantifier into the lexical entry for each verb and give it scope over the variable f 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:

(52) a. *bark* $\rightsquigarrow \lambda f \exists e.\mathsf{bark}(e) \wedge f(e)$
b. *butter* $\rightsquigarrow \lambda f \exists e.\mathsf{butter}(e) \wedge f(e)$
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 $\langle\langle v, t\rangle, t\rangle$. For this reason, we will no longer rely on predicate modification, but instead use function application to combine syntactic arguments with verbal projections. This means that our thematic roles look more complicated than before:

(53) a. *[agent]* $\rightsquigarrow \lambda x \lambda V \lambda f.V(\lambda e.\mathsf{agent}(e) = x \wedge f(e))$
b. *[theme]* $\rightsquigarrow \lambda x \lambda V \lambda f.V(\lambda e.\mathsf{theme}(e) = x \wedge f(e))$
c. ...

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 f so long as f 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, $\lambda e.\mathsf{true}$, contains such an event:

(54) a. $\lambda f \exists e[\mathsf{bark}(e) \wedge \mathsf{ag}(e) = \mathsf{s} \wedge f(e)](\lambda e.\mathsf{true})$
b. $\exists e[\mathsf{bark}(e) \wedge \mathsf{ag}(e) = \mathsf{s} \wedge (\lambda e.\mathsf{true})(e)]$
c. $\exists e[\mathsf{bark}(e) \wedge \mathsf{ag}(e) = \mathsf{s} \wedge \mathsf{true}]$
d. $\exists e[\mathsf{bark}(e) \wedge \mathsf{ag}(e) = \mathsf{s}]$

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:

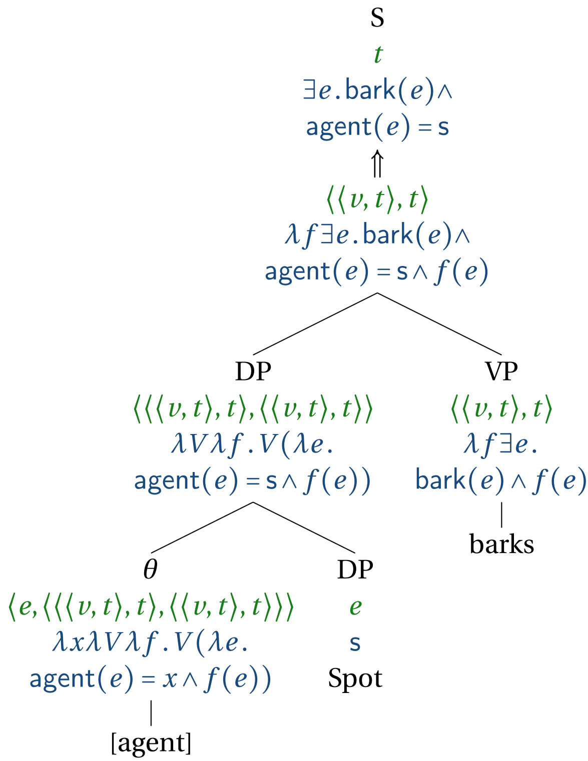
Type-Shifting Rule 6. Quantifier Closure
If $\alpha \rightsquigarrow \alpha'$, where α' is of category $\langle\langle v,t\rangle,t\rangle$, then:

$$\alpha \rightsquigarrow \alpha'(\lambda e.\mathsf{true})$$

as well.

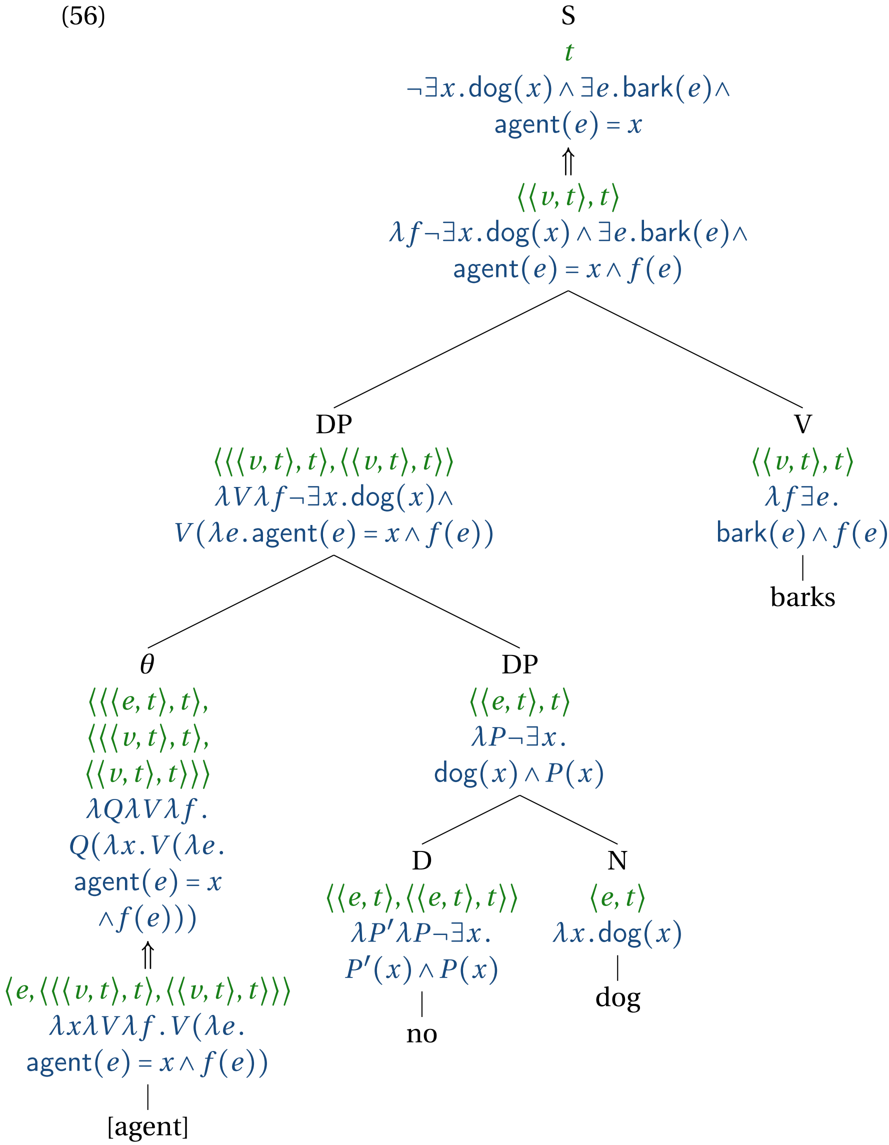
The full derivation of the sentence is shown in (55).

(55)



We are now ready to interpret a quantificational noun phrase. This time, applying Hendriks' raising schema to the theta role gives the right result, as shown in (56). We do not need to apply quantifier raising. This is as expected, because the quantifier is contained in the entry for the verb, so the subject already takes syntactic scope over it.

(56)



Let us now see how syntactic adjuncts, such as adverbs, are treated on this approach. Just like syntactic arguments, adjuncts are combined with verbal projections using Function Application instead of Predicate Modification. This makes the representations of adverbs more complicated:

(57) a. *slowly* $\rightsquigarrow \lambda V \lambda f . V(\lambda e.\mathsf{slow}(e) \wedge f(e))$
b. ...

An example of a derivation that uses this adverb is shown in (58). To save space, the VP *buttered the toast* is shown as a unit, and as before, we pretend that *the toast* is a constant rather than a definite description. Nothing of consequence would change if we didn't.

(58)

- S: t, $\exists e.\mathsf{agent}(e) = \mathsf{j} \wedge \mathsf{butter}(e) \wedge \mathsf{theme}(e) = \mathsf{t} \wedge \mathsf{slow}(e)$
  - $\Uparrow$ $\langle\langle v,t\rangle,t\rangle$, $\lambda f \exists e.\mathsf{agent}(e) = \mathsf{j} \wedge \mathsf{butter}(e) \wedge \mathsf{theme}(e) = \mathsf{t} \wedge \mathsf{slow}(e) \wedge f(e)$
    - DP: $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle$, $\lambda V \lambda f.V(\lambda e.\mathsf{agent}(e) = \mathsf{j} \wedge f(e))$
      - θ: $\langle e,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$, $\lambda x \lambda V \lambda f.V(\lambda e.\mathsf{agent}(e) = x \wedge f(e))$
        - [agent]
      - DP: e, j
        - Jones
    - VP: $\langle\langle v,t\rangle,t\rangle$, $\lambda f \exists e.\mathsf{butter}(e) \wedge \mathsf{theme}(e) = \mathsf{t} \wedge \mathsf{slow}(e) \wedge f(e)$
      - V: $\langle\langle v,t\rangle,t\rangle$, $\lambda f \exists e.\mathsf{butter}(e) \wedge \mathsf{theme}(e) = \mathsf{t} \wedge f(e)$
        - buttered the toast
      - AdvP: $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle$, $\lambda V \lambda f.V(\lambda e.\mathsf{slow}(e) \wedge f(e))$
        - slowly

From what we have seen so far, the choice between the two approaches depends mainly on whether the preferred way to deal with quantificational noun phrases is by quantifier raising or type shifting. The next sections compare the two systems with respect to two other phenomena, conjunction and negation.

11.4.2 Another formal fragment

Let us recapitulate the additions to our fragment.The syntax is defined as in three-valued type logic (L_λ with three truth values as in Chapter 8), plus the following additions:

Syntax rules. We add the following rule:

(59) **Syntax**
DP $\rightarrow$ θ DP

Lexicon. Lexical items are associated with syntactic categories as follows:

$$\theta: \quad \text{[agent], [theme], } \ldots$$

Translations. Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- e is a variable of type v
- f is a variable of type $\langle v, t\rangle$
- V is a variable of type $\langle\langle v, t\rangle, t\rangle$
- bark and butter are constants of type $\langle v, t\rangle$
- agent and theme are constants of type $\langle v, e\rangle$

The following entries replace the previous ones:

Type $\langle v, t\rangle$:

1. *bark* $\rightsquigarrow \lambda f \exists e . \mathsf{bark}(e) \wedge f(e)$
2. *butter* $\rightsquigarrow \lambda f \exists e . \mathsf{butter}(e) \wedge f(e)$

Type $\langle v, e\rangle$:

1. *[agent]* $\rightsquigarrow \lambda x \lambda V \lambda f \,.\, V(\lambda e.\mathsf{agent}(e) = x \wedge f(e))$
2. *[theme]* $\rightsquigarrow \lambda x \lambda V \lambda f \,.\, V(\lambda e.\mathsf{theme}(e) = x \wedge f(e))$

Type $\langle\langle\langle v,t\rangle,t\rangle,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$:

1. $\text{and}_{\text{VP}} \rightsquigarrow \lambda V' \lambda V \lambda f \,.\, V(f) \wedge V'(f)$

We have introduced the following type-shifter:

Type-Shifting Rule 7. Quantifier Closure
If $\alpha \rightsquigarrow \alpha'$, where α' is of type $\langle\langle v,t\rangle,t\rangle$, then:

$$\alpha \rightsquigarrow \alpha'(\lambda e.\mathsf{true})$$

as well.

11.5 Conjunction in event semantics

In Chapter 10, 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:

(60) $$\text{and}_{\langle\tau,\langle\tau,\tau\rangle\rangle} \rightsquigarrow \begin{cases} \lambda q \lambda p \,.\, p \wedge q & \text{if } \tau = t \\ \lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1} \,.\, \langle\!\langle \text{and} \rangle\!\rangle_{\langle\sigma_2,\langle\sigma_2,\sigma_2\rangle\rangle}(X(Z))(Y(Z)) & \text{if } \tau = \langle\sigma_1,\sigma_2\rangle \end{cases}$$

where $\langle\!\langle \text{and} \rangle\!\rangle_{\langle\sigma_2,\langle\sigma_2,\sigma_2\rangle\rangle}$ denotes the translation of *and* for the corresponding type.

What does this rule amount to in the case of VP-modifying *and*, as in *John smoked and drank*? On the first approach, VPs are of type $\tau = \langle v,t\rangle$. On the second approach, VPs are of type $\tau = \langle\langle v,t\rangle,t\rangle$. Applying rule (60) in each case results in the following:

(61) a. and$_{\text{VP}} \rightsquigarrow \lambda f' \lambda f \lambda e.\, f(e) \wedge f'(e)$
b. and$_{\text{VP}} \rightsquigarrow \lambda V' \lambda V \lambda f.\, V(f) \wedge V'(f)$

Exercise 2. Show how rule (60) leads to these two representations.

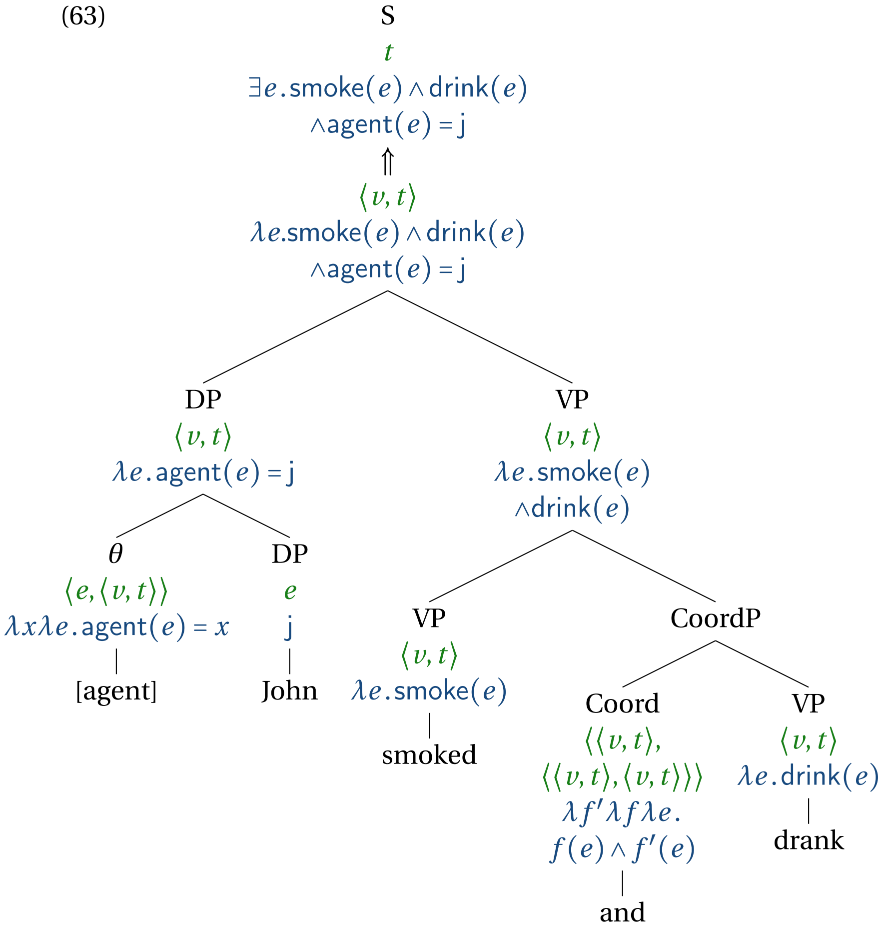
As you can see in (63) and (64), these two choices lead to very different translations: (62a) and (62b) respectively.

(62) a. $\exists e.\mathsf{smoke}(e) \wedge \mathsf{drink}(e) \wedge \mathsf{agent}(e) = \mathsf{j}$
b. $\exists e.\mathsf{smoke}(e) \wedge \mathsf{agent}(e) = \mathsf{j} \wedge$
$\exists e'.\mathsf{drink}(e') \wedge \mathsf{agent}(e') = \mathsf{j}$

Now, (62a) 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 (62a) 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 (62a) sends us right back to square one. We fare much better with (62b), because it provides us with the two events we need to avoid the problem.

Exercise 3. Add *slowly* and *quickly* to the tree in (64) and show how the resulting formula avoids the attribution of contradictory properties to the same event.

(63)


S
t
$\exists e.\mathsf{smoke}(e) \wedge \mathsf{drink}(e)$
$\wedge \mathsf{agent}(e) = \mathsf{j}$
$\Uparrow$
$\langle v, t\rangle$
$\lambda e.\mathsf{smoke}(e) \wedge \mathsf{drink}(e)$
$\wedge \mathsf{agent}(e) = \mathsf{j}$
DP
$\langle v, t\rangle$
$\lambda e.\mathsf{agent}(e) = \mathsf{j}$
θ
$\langle e, \langle v, t\rangle\rangle$
$\lambda x \lambda e.\mathsf{agent}(e) = x$
[agent]
DP
e
j
John
VP
$\langle v, t\rangle$
$\lambda e.\mathsf{smoke}(e)$
$\wedge \mathsf{drink}(e)$
VP
$\langle v, t\rangle$
$\lambda e.\mathsf{smoke}(e)$
smoked
CoordP
Coord
$\langle\langle v, t\rangle,$
$\langle\langle v, t\rangle, \langle v, t\rangle\rangle\rangle$
$\lambda f' \lambda f \lambda e.$
$f(e) \wedge f'(e)$
and
VP
$\langle v, t\rangle$
$\lambda e.\mathsf{drink}(e)$
drank

(64)

- S
  t
  $\exists e.\text{smoke}(e) \wedge \text{agent}(e) = \text{j} \wedge \exists e'.\text{drink}(e') \wedge \text{agent}(e') = \text{j}$
  $\Uparrow$
  $\langle\langle v, t\rangle, t\rangle$
  $\lambda f \exists e.\text{smoke}(e) \wedge \text{agent}(e) = \text{j} \wedge f(e) \wedge \exists e'.\text{drink}(e') \wedge \text{agent}(e') = \text{j} \wedge f(e)$
  - DP
    $\langle\langle\langle v, t\rangle, t\rangle, \langle\langle v, t\rangle, t\rangle\rangle$
    $\lambda V \lambda f.V(\lambda e.\text{agent}(e) = \text{j} \wedge f(e))$
    - θ
      $\langle e, \langle\langle\langle v, t\rangle, t\rangle, \langle\langle v, t\rangle, t\rangle\rangle\rangle$
      $\lambda x \lambda V \lambda f.V(\lambda e.\text{agent}(e) = x \wedge f(e))$
      - [agent]
    - DP
      e
      j
      - John
  - VP
    $\langle\langle v, t\rangle, t\rangle$
    $\lambda f \exists e.\text{smoke}(e) \wedge f(e) \wedge \exists e'.\text{drink}(e') \wedge f(e')$
    - VP
      $\langle\langle v, t\rangle, t\rangle$
      $\lambda f \exists e.\text{smoke}(e) \wedge f(e)$
      - smoked
    - CoordP
      - Coord
        $\langle\langle\langle v, t\rangle, t\rangle, \langle\langle\langle v, t\rangle, t\rangle, \langle\langle v, t\rangle, t\rangle\rangle\rangle$
        $\lambda V' \lambda V \lambda f.V(f) \wedge V'(f)$
        - and
      - VP
        $\langle\langle v, t\rangle, t\rangle$
        $\lambda f \exists e'.\text{drink}(e') \wedge f(e')$
        - drank

Does this mean that we cannot represent conjunction on the first approach? No: all we have seen is that the Partee & Rooth schema is not compatible with it. 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 10 we have encountered both schema-based and non-schema-based entries.

Exercise 4. Formulate an entry for VP-level conjunction that is compatible with the event-predicate based 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, $\theta(e \oplus e') = \theta(e) \oplus \theta(e')$.

11.6 Negation in event semantics

Quantifiers and coordination are scope-taking elements whose behavior with respect to events we need to think about. Negation is another. Just like quantificational noun phrases, negation always seems to take scope above the event quantifier. For example, (65), read with neutral intonation, only has the reading in (66b), and lacks the reading in (67b). That reading, if it was available, would be almost trivially true, since any event that doesn't happen to be a barking by Spot will verify it.

(65) Spot didn't bark.

(66) a. $\neg[\exists e . \mathsf{bark}(e) \wedge \mathsf{agent}(e) = s]$
b. "There is no barking event that is done by Spot"

(67) a. $\exists e . \neg[\mathsf{bark}(e) \wedge \mathsf{agent}(e) = s]$
b. "There is an event that is not a barking by Spot"

How do the two approaches to event semantics that we have encountered fare? Let us start with the first approach, on which verbs denote sets of events. On this approach, verbs and their projections denote sets of events. For example, the verb *bark* denotes the set of all barking events. And so does the VP, on the assumption that it only consists of this verb. Now VP negation needs to map this set to another set of events. What could that set be? If VP negation is translated in terms of truth-functional negation (that is, the kind of negation that we are familiar with from propositional logic and predicate logic), we might attempt this:

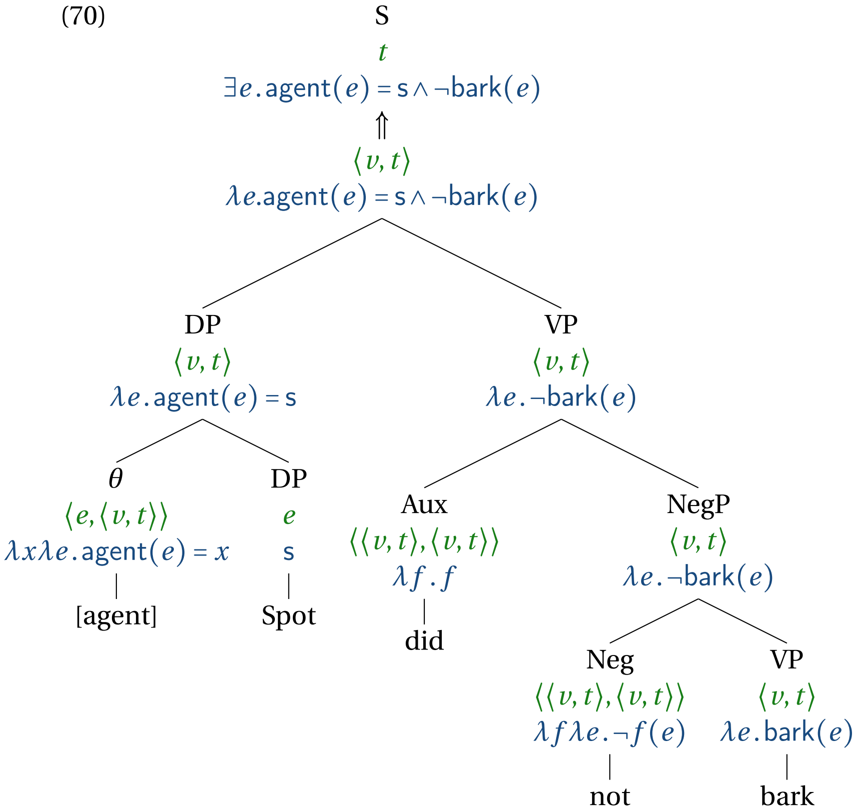
(68) not $\rightsquigarrow \lambda f \lambda e . \neg f(e)$ (to be revised)

But this is a disastrous denotation. It says that *not* applies to a set of events and maps it to its complement. For example, if it applies to the set denoted by *bark*, the result will be the complement of the set of barking events. The subject then combines with this set via a thematic role head, and the result asserts that there is an event whose agent is Spot that is not a barking event. This derivation is shown in Figure (70). The result is the following reading:

(69) $\exists e . \mathsf{agent}(e) = \mathsf{s} \wedge \neg \mathsf{bark}(e)$
There is an event whose agent is Spot that is not a barking event.

What we want is the reading expressed by (66b). But the entry in (68) generates (69) instead, which expresses something much weaker than what we want. Formula (69) is true just in case Spot did anything at all instead of or in addition to barking. The problem runs deeper than the faulty translation in (68). 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.

(70)

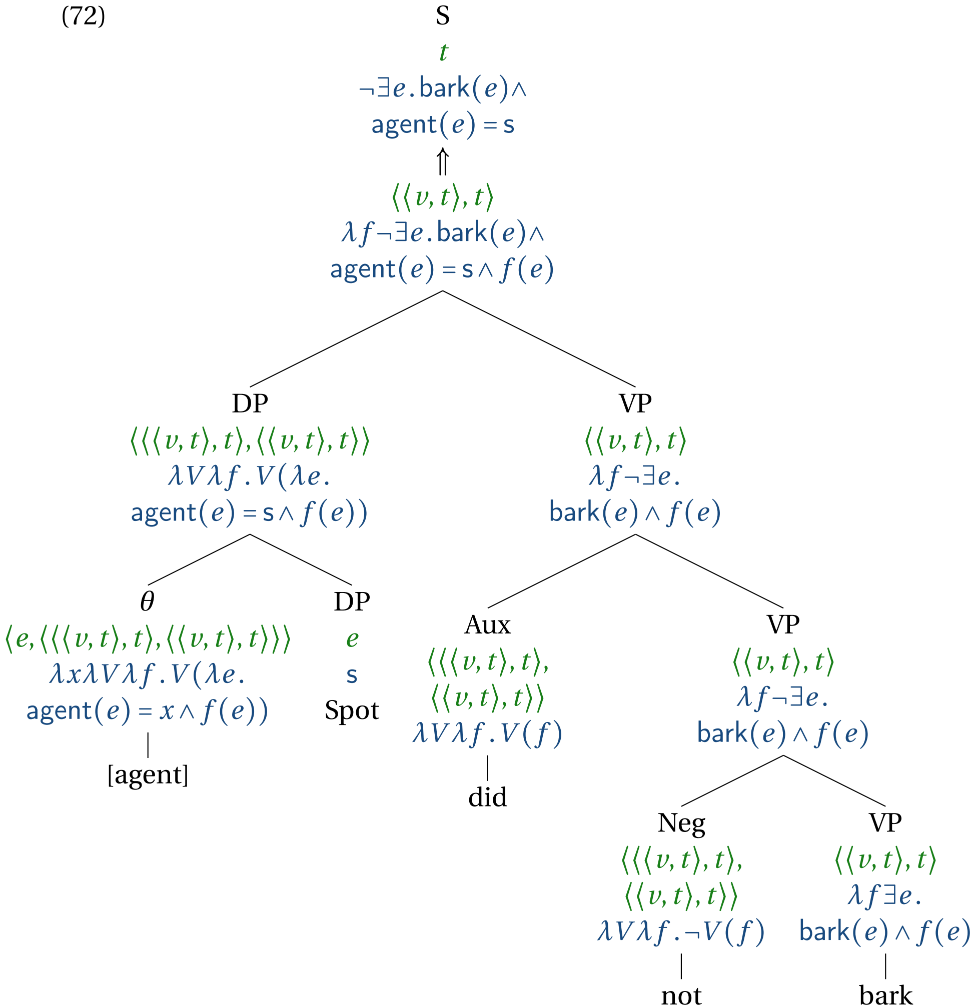


One can respond to this situation in different ways. One way is to expand our inventory of events to include "negative events". In some cases, it is intuitively clear what a negative event should be: for example, a negative staying event is a leaving event and vice versa. While it is not so clear what a negative barking event is, some semanticists have tried to clarify their status (Bernard & Champollion, 2018). Another way is to include the subject into the VP (see the discussion of the VP-internal subject hypothesis in Chapter 7), so that *not* applies to *Spot did bark* rather than to *bark* and returns a truth value rather than a set of events. (Similarly, *not* could take a VP and a subject and combine them to return a truth value.) But this will not extend easily to sentences in which event modifiers like *in the garden* take scope over the subject, as in *In the garden, Spot didn't bark*, because such event modifiers expect to be given sets of events. The tree for this sentence would look like (70) except that the lower half of the S node would combine via Predicate Modification with a PP node with type $\langle\langle v, t\rangle, t\rangle$ and denotation $\lambda e.\text{location}(e) = \iota x.\text{Garden}(x)$ and then the result (after existential closure applies) would be the following:

(71) $\exists e.\text{location}(e) = \iota x.\text{Garden}(x) \wedge \text{agent}(e) = \text{s} \wedge \neg\text{bark}(e)$

We will not implement any of these options in detail and instead adopt the second approach to event semantics presented in this chapter, on which verbs and their projections denote sets of sets of events. On this approach, *not bark* can be given a straightforward denotation: the set of sets that do not contain any barking events. The resulting truth conditions are the desired ones in (66b). This is shown in Figure (72).

(72)


S
t
$\neg\exists e.\mathsf{bark}(e)\wedge$
$\mathsf{agent}(e)=\mathsf{s}$
$\Uparrow$
$\langle\langle v,t\rangle,t\rangle$
$\lambda f\neg\exists e.\mathsf{bark}(e)\wedge$
$\mathsf{agent}(e)=\mathsf{s}\wedge f(e)$
DP
$\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle$
$\lambda V\lambda f.V(\lambda e.$
$\mathsf{agent}(e)=\mathsf{s}\wedge f(e))$
VP
$\langle\langle v,t\rangle,t\rangle$
$\lambda f\neg\exists e.$
$\mathsf{bark}(e)\wedge f(e)$
θ
$\langle e,\langle\langle\langle v,t\rangle,t\rangle,\langle\langle v,t\rangle,t\rangle\rangle\rangle$
$\lambda x\lambda V\lambda f.V(\lambda e.$
$\mathsf{agent}(e)=x\wedge f(e))$
[agent]
DP
e
s
Spot
Aux
$\langle\langle\langle v,t\rangle,t\rangle,$
$\langle\langle v,t\rangle,t\rangle\rangle$
$\lambda V\lambda f.V(f)$
did
VP
$\langle\langle v,t\rangle,t\rangle$
$\lambda f\neg\exists e.$
$\mathsf{bark}(e)\wedge f(e)$
Neg
$\langle\langle\langle v,t\rangle,t\rangle,$
$\langle\langle v,t\rangle,t\rangle\rangle$
$\lambda V\lambda f.\neg V(f)$
not
VP
$\langle\langle v,t\rangle,t\rangle$
$\lambda f\exists e.$
$\mathsf{bark}(e)\wedge f(e)$
bark

12 Tense and aspect

12.1 Introduction

12.1.1 Temporalism vs. eternalism

Some sentences appear to change in 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:

(1) The queen is ill.

Does the status of a sentence as true or false vary over time? In philosophy, that stance is known as TEMPORALISM, and it is an idea that Arthur Prior formalized with his TENSE LOGIC. In Priorian tense logic, a sentence may be true relative to one time but false relative to another. The logic includes operators like 'Future' and 'Past' that can shift the time of evaluation. For example, 'Future ϕ' is true at time t if there is a time t' after t such that ϕ is true at t.

The view on which the truth status of sentences does *not* change over time is called ETERNALISM. An eternalist could account for the apparent change in truth value from one moment to another by saying that the present tense in (1) is a deictic element, so the sentence cannot even be evaluated as true or false until the context of utterance fills in a value for it. Just like with other sentences containing indexicals, such as *I am here now,* the proposition expressed by the sentence depends on information about the context of utterance. That proposition is a claim about a particular time, such as 2pm on Thursday, August 10th, 2024. If the queen is ill at that time, then even if the queen recovers by August 11, it will still be true true on August 11 that the queen was ill at 2pm on August 10th, and that time-specific claim will remain true for eternity.

Kaplan's distinction between CONTENT and CHARACTER is helpful in articulating the sense in which the truth of the proposition expressed in (1) can coherently be thought of as stable across time.

- 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.

For illustration, consider the following two utterances:

(2) 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.

Do these two sentences have the same meaning or different meaning? Reasonable people might disagree. How about the following pair:

(3) 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. In some sense, the examples in (2) have the same meaning, but in another sense, those in (3) have the same meaning. The pair of sentences in (2) have the same character, while the pair in (3) have the same content.

The character of a sentence can be modeled as a function from contexts to contents that fills in values for all of the indexicals; the content is then a claim that is free of context-sensitivity.

In Kaplan's terms, the content of an utterance of (1), in any given context, is a claim about a particular time (the time of utterance in that context). The truth value of that claim does not change over time. If the present tense is a deictic element, then eternalism is a coherent stance to take, at the level of content.

12.1.2 Some desiderata for a theory of tense

12.1.2.1 The deictic nature of tense

There are several reasons to reject the view of tense that is encoded in Prior's tense logic. Unlike in Prior's tense logic, natural languages do not seem to have operators that shift the time of evaluation, like Prior's 'Past' and 'Future' operators. 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).

(4) 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.

(5) 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 (6), which can:

(6) Fred told me that Mary was pregnant.

Both (4) and (5) show a tendency for the present tense to be anchored to the time of utterance.

The interpretation of future tense is similarly impervious to embedding:

(7) It was predicted that the Messiah will come.

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 (7) with *would* in place of *will*.) 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.

12.1.2.2 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 (8) describes an event that directly follows the event described in the first sentence (Kamp & Reyle, 1993, ex. (5.21), p. 495).

(8) 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 (8) 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:

(9) 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 (9) 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 (9) 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:

(10) 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:

(11) Whenever I visited Carla, she was lying in bed.

One might even go so far as to posit that the variable in question is in fact *always* bound by a quantifier. It is commonly assumed that the domain of quantifiers is restricted by a contextually supplied argument, often thought of as similar to a pronoun (von Fintel, 1994). This kind of contextual domain restriction can make a quantificational analysis of tenses viable. On such an analysis, (9) is literally false with respect to the entire domain, but true with respect to a narrower domain which only includes contextually relevant times. In this chapter, we will assume that the past tense is interpreted as a variable over times that may be either free or bound.

12.1.2.3 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').[1] 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.

[1] Here we are *not* primarily concerned with the various aspectual classes most famously laid out by Vendler (1957) (state, activity, accomplishment, achievement, and other distinctions of this kind). Those types of distinctions fall under the heading of AKTIONSART (type of event), also known as 'situation aspect', 'lexical aspect', or 'inner 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). 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.

(12) 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.'

(13) 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 (12b) and (13b) show, the past progressive (*was V-ing*) only captures one of the meanings that the past imperfective can have in French and Russian. 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. Furthermore, in English, progressive forms are used to indicate imperfectivity in combination with dynamic (non-stative) predicates, but unlike imperfective in languages like French and Russian, the progres-

	PERFECTIVE	IMPERFECTIVE
PERFECT	*I have danced*	*I have been dancing*
NON-PERFECT	*I danced*	*I was dancing*

Table 12.1: Aspectual distinctions in English

sive is not regularly used with stative verbs, as mentioned above in the discussion of aktionsart. So we follow Comrie (1976, 25) in treating 'progressive' as a distinct sub-category of imperfective aspect.

English has two morphological forms that have been thought to express viewpoint aspect: the progressive, as in *I am eating* the perfect, as in *I have eaten*. 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 12.1. It is controversial whether the perfect counts as a form of viewpoint aspect; Bhatt & Pancheva (2005) argue that it is not, contrary to the more common view. In any case, it is a grammatical category that can combine with other tense/aspect forms to indicate temporal relations, and plays an important role in the theory of tense. We turn to it next.

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)

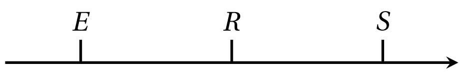
All three of these sentences describe an event of seeing something that precedes the time of utterance, but they all differ in meaning.

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).

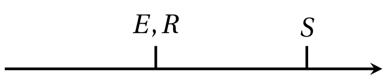
- SPEECH TIME (S): the time the sentence is uttered
- EVENT TIME (E): the time the event takes place
- REFERENCE TIME (R): the time under discussion

The concept of ‘reference time’ was Reichenbach’s major innovation, and the concept that is least intuitively obvious. 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.

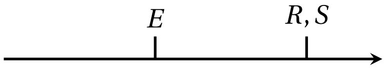


Aside from its intuitive appeal, support for this analysis comes from the fact that temporal adverbs like *at 5pm* track the hypothesized reference time:

(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 1. With the tools just developed, explain the following contrast, discussed by Reichenbach.

(21) How unfortunate!

a. Now that John tells me this I have mailed the letter.

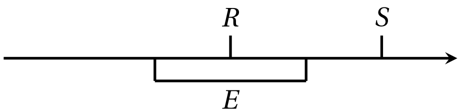
b. #Now that John tells me this I mailed the letter.

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. There are a number of facts about the English perfect that the Reichenbachian analysis fails to account for (see Bhatt & Pancheva 2005 for an excellent overview and discussion), but let us stop here for the moment and move on to another grammatical device for the encoding of temporal relations in English, the progressive.

The progressive Consider the following contrast:

(22) a. At 5pm, I mailed the letter. (simple past)
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 (22b), the time interval during which the dancing event took place includes the reference time identified with *5pm*. This gives us the following picture for past progressive:



Following Comrie (1976), we take the unmarked, non-progressive variant to have PERFECTIVE ASPECT in English. Comrie considers the English progressive to be a subtype of IMPERFECTIVE ASPECT, a category of grammatical forms associated with a lack of completion, a continuous state, an iterated sequence of events, or a habitual pattern.

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 (22b); 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.

(23) a. I walked to the park $\Rightarrow$ I got to the park.
  b. I was walking to the park $\not\Rightarrow$ 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 (1990b) and Dowty (1979) point out, this analysis predicts that the following argument should be valid:

(24) Mary was building a house. Therefore, Mary will have built a house.

Another fact to account for is that progressives behave like atelic predicates:

(25) a. I walked to the park in/*for an hour.
  b. I was walking to the park for/*in an hour.

There are many more interesting observations to consider, and many approaches to the analysis of the progressive; see Bhatt &

Pancheva (2005), lecture 4 for an overview. For simplicity, we will stick with the Bennett and Partee analysis, acknowledging that there is much more to the story.

12.2 A formal theory of tense

12.2.1 A partial, context-sensitive logic with times

12.2.1.1 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. The language contains an infinite set of variables of type i, all of the form t_n, where n is an integer. We use t' as an abbreviation for t_1, etc.

L_T contains the symbols $<$ and $\subseteq$, standing for temporal precedence and temporal inclusion:

(26) Syntactic rule ($<$)
If α and β are expressions of type i, then $\alpha < \beta$ is a formula.

(27) 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 12.2.1.2.)

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:

(28) $\textsf{ill}(x)(t)$

(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:

(29) $\mathsf{ill}_t(x)$

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:

(30) $\mathsf{ill}_{\mathsf{now}}(\mathsf{q})$

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 the queen being ill", 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:

(31) $\mathsf{ill}_t(\mathsf{q})$

to express the existential claim that there was a state of the queen being ill whose full temporal extent was t. We are not explicitly appealing to eventualities in this chapter, but if we were, we would write (31) as:

(32) $\exists e.\mathsf{ill}(e) \wedge \mathsf{holder}(e, \mathsf{q}) \wedge \tau(e) = t$

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.

12.2.1.2 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 Chapter 7, where the semantic value of an expression is determined relative to a model M, an assignment function g, a world w, 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:

- $D^+_t = (D_t \cup \{\#_t\})$.
- $D^+_e = (D_t \cup \{\#_e\})$.
- $D^+_i = (D_i \cup \{\#_i\})$.

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 c such that $t(c) \in T$, $sp(c) \in D$, and $ad(c) \in D$.

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.[2] 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:

(33) $$[\![\alpha < \beta]\!]^{M,g,c} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g,c} < [\![\beta]\!]^{M,g,c} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

(34) $$[\![\alpha \subseteq \beta]\!]^{M,g,c} = \begin{cases} \mathsf{T} & \text{if } [\![\alpha]\!]^{M,g,c} \subseteq [\![\beta]\!]^{M,g,c} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

The special indexical constant now is interpreted relative to a given context of utterance c:

(35) $$[\![\mathsf{now}]\!]^{M,g,c} = t(c)$$

[2] 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).

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 $g(x)$:

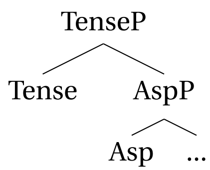
(36) $[\![\mathsf{ill}_{\mathsf{now}}(x)]\!]^{M,g,c} = \mathsf{T}$ iff $I(\mathsf{ill})(t(c))(g(x))$

12.2.2 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.

12.2.2.1 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:

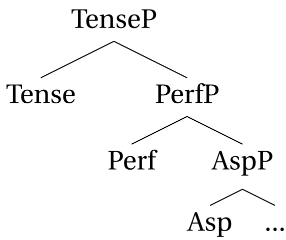
(37)


If perfect is considered a form of viewpoint aspect, then this structure captures the fact that a tense form can combine with either perfect and progressive. However, it does not take into account the fact that the perfect and the progressive can be combined:

(38)
a. I have been reading it.
b. I had been reading it.
c. I will have been reading it.

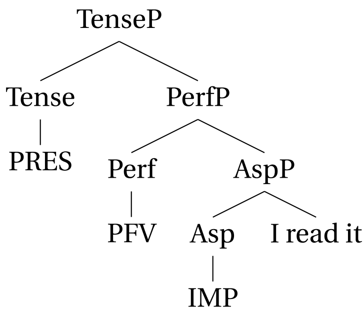
We will therefore assume that there is a syntactic layer for the perfect above the aspectual layer where the perfective/imperfective distinction is encoded (McCoard, 1979; Dowty, 1979; Iatridou et al., 2001; Bhatt & Pancheva, 2005).

(39)



Thus (38a) has the structure:

(40)

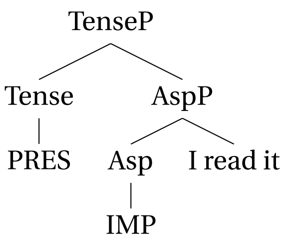


That said, we assume that the PerfP projection is optional, so a non-perfect sentence like:

(41) I am reading it.

has the structure:

(42)



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.

12.2.2.2 Non-finite clauses

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:

(43) *dance* $\rightsquigarrow \lambda x.\lambda t.\mathsf{dance}_t(x)$

(44) *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:

(45) $\lambda t.\mathsf{read}_t(\mathsf{a}, x_i)$

12.2.2.3 Viewpoint aspect

Recall the contrasting pair of examples (22), repeated here:

(46) a. At 5pm, I mailed the letter. (simple past)
b. At 5pm, I was mailing the letter. (past progressive)

Let us assume, following Bennett & Partee (1972), that the English progressive form signals that the event time includes the reference time.

Syntactically, we analyze (46b) as follows:

(47) At 5pm [$_{\text{TP}}$ PAST [$_{\text{AspP}}$ PROG [$_{\text{vP}}$ I mail the letter]]]

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 Asp node 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.

(48) 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 $\underline{t \text{ is contained in a time } t'}$ at which P is true.'

We also treat perfective aspect in terms of inclusion, but in the other direction:

(49) 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 predicate of times that is true of a time t if $\underline{t \text{ contains a time } t'}$ 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 AspP node of (46b):

(50) $[_{\text{AspP}}$ PROG $[_{\text{vP}}$ 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' containing t at which the speaker mails the letter.

We assume that non-progressive sentences are interpreted as perfective. Hence the AspP node of (46a) would be translated as follows:

(51) $[_{\text{AspP}}$ 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.

12.2.2.4 Tense

Let us now outline a simple theory of tense.

Past tense. Following Partee and others, we assume that the past tense contributes a free variable over times. We assume that the natural language morpheme PAST is associated with an index n, just like a pronoun. This index determines the variable over times that the past tense morpheme maps to. The simplest possible analysis implementing this idea would be the following:

(52) $\text{PAST}_n \rightsquigarrow t_n$ (first version)

Then *I mailed the letter* would be analyzed as follows (where 3 is an arbitrary index):

(53) $[_{\text{TP}}$ PAST3 $[_{\text{AspP}}$ PFV $[_{\text{vP}}$ I mail the letter]]]
$\exists t'[t' \subseteq t_3 \wedge \text{mail}_{t'}(\text{i}, \iota x.\text{letter}(x))]$

The formula we derive expresses the claim there is a time t' contained within the discourse-salient time t_3 at which the speaker mails the letter.

One small deficiency of the analysis in (52), of course, is that it says nothing about how the time relates to the time of utterance. Nothing constrains t_3, for example, to be in the past. Following Heim (1994), we implement this constraint as a presupposition, just like gender features on pronouns. (If the constraint were an entailment, then it should be possible to target the constraint with negation, and *I didn't turn off the stove* could be true in virtue of there being a non-past time at which the speaker turns off the stove.)

Another question that arises under the simple treatment we started with in (52) is how modifiers like *at 5pm* get integrated

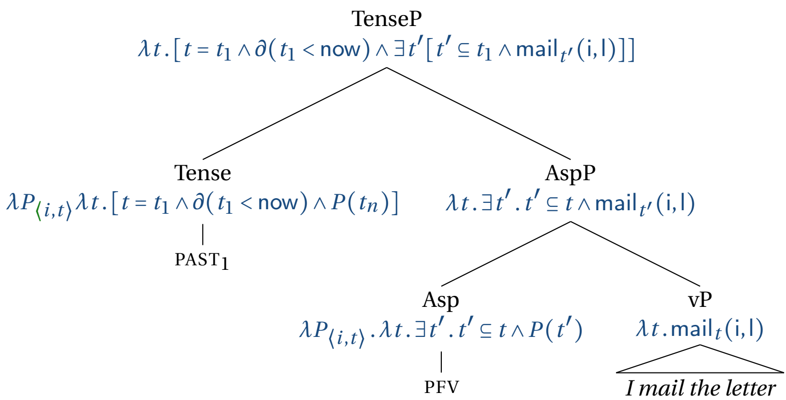


Figure 12.1: Derivation for *I mailed the letter.*

compositionally into the sentence. These modifiers determine the identity of the reference time. To handle this sort of phenomenon, we assume that tenses contribute an argument slot that can be saturated by a time-denoting expression. Our final lexical entry for the past tense morpheme is as follows.

(54) PAST$_n$ $\rightsquigarrow \lambda P_{\langle i,t\rangle}\lambda t.[t = t_n \wedge \partial(t_n < \mathsf{now}) \wedge P(t_n)]$ (final)

Thus for an example like *I mailed the letter*, we have the derivation in Figure 12.1. At the top node, t_1 is a free variable over times that is presupposed to precede the moment of speech. The discourse context should provide an assignment function that will give a value to this free variable. As long as the value is one that precedes the time of utterance, the sentence will have a defined truth value. In other words, the expression will have 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 2. Write out how you would read the expression at the top of Figure 12.1 aloud.

Let us assume that a modifier like *at 5pm* denotes a particular time interval, and let us use $\mathsf{5pm}$ as a constant of type i ot represent it. (In fact, there are many times called *5pm*, and there is some pragmatic work to be done in identifying the one in question, but we will gloss over that subtlety in the present discussion.) Then *I mailed the letter at 5pm* can be translated with the following formula:

(55) at 5pm [$_{\text{TP}}$ PAST3 [$_{\text{AspP}}$ PFV [$_{\text{vP}}$ I mail the letter]]]
$[\mathsf{5pm} = t_1 \wedge \partial(t_1 < \mathsf{now}) \wedge \exists t'[t' \subseteq t_1 \wedge \mathsf{mail}_{t'}(\mathsf{i}, \mathsf{l})]]$

In this formula, the variable t_1 is still free, but it is identified with 5pm, so there is only one value that an assignment function can give it that will cause the formula to be true.

In the case that there is no temporal modifier, let us assume that the variable t is existentially bound. To implement this, let us assume that a silent existential quantifier may appear at LF. We'll notate it as ∃, mnemonically. With this silent existential quantifier at LF, *I mailed the letter* receives the translation:

(56) ∃ [$_{\text{TP}}$ PAST3 [$_{\text{AspP}}$ PFV [$_{\text{vP}}$ I mail the letter]]]
$\exists t[t = t_1 \wedge \partial(t_1 < \mathsf{now}) \wedge \exists t'[t' \subseteq t_1 \wedge \mathsf{mail}_{t'}(\mathsf{i}, \mathsf{l})]]$

Although t is existentially bound here, it is identified with a free variable, so our treatment is still a referential theory of tense, rather than an existential one.

Exercise 3. 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 4. Compute a tree for *I was mailing the letter at 5pm.* Does the formula you derive carry a completion inference? Explain why or why not.

Present tense. A simple analysis of the present tense would be as follows:

(57) PRES $\rightsquigarrow$ $\textsf{now}$

Then *I am mailing the letter* would have the following translation:

(58) [$_{\text{TP}}$ PRES [$_{\text{AspP}}$ PROG [$_{\text{vP}}$ I mail the letter]]]
$\exists t'[\textsf{now} \subseteq t' \wedge \textsf{mail}_{t'}(\textsf{i}, \textsf{l})]$

This formula asserts that there is a time t' including the time of utterance at which a letter-mailing event is taking place. The event need not be complete by the time of utterance; the event is merely in progress at the time of utterance.

This treatment does not straightforwardly allow for overt temporal modifiers, as in *Now I am mailing the letter.* Granted, the distribution of present tense with temporal modifiers is somewhat limited; the following sentence is slightly odd:

(59) ?At 5pm, I am mailing the letter.

The problem seems to be that the use of *at 5pm* as a temporal modifier carries a pragmatic inference of some kind that 5pm is distinct from the time of utterance. But in principle, there is no ban on the use of temporal modifiers with the present tense. Indeed, one of the important facts that we aim to capture in this chapter is the contrast between (17) on the one hand and (18a) and (18b) on the other. We repeat the examples here:

(60) Now I have mailed the letter.

(61) a. ??Now I had mailed the letter.
b. ??Now I mailed the letter.

Once we develop an analysis of the English perfect (see section 12.2.2.5, we will be able to explain this contrast. For now, let us modify our lexical entry for the present tense to allow for direct semantic composition with temporal modifiers:

(62) PRES $\rightsquigarrow \lambda P_{\langle i,t \rangle} \lambda t . [t = \mathsf{now} \land P(t)]$

In order to get an expression of type t at the top node of a tree for a present tense sentence, a silent existential quantifier can be inserted. In that case, our analysis of the present tense boils down to the simple analysis we started with, using the lexical entry in (57). Hence, when there are no modifiers, it suffices to use that simple analysis.

Exercise 5. Give a derivation tree for *I am mailing the letter.* Use a null existential quantifier over times in order to get a formula at the top node.

Now, in English, the simple present tense cannot be used with dynamic predicates to describe an event that is currently happening:

(63) 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. One possible way of accounting for the awkwardness of (63) would be to assume that in English, the simple present contains perfective aspect, and that the time of utterance is a mere instant of time. There is plenty more to say about this issue, but we will leave it at that for the moment.

Exercise 6. 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 (63)? 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:

(64) 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.[3] We will maintain the simple theory of the present tense in (62) 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

[3] 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, because the sentence can mean 'I am the only person x such that x did x's homework', not 'I am the only person x such that x did my (the speaker's) homework.'

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:

(65) FUT $\rightsquigarrow \lambda P_{\langle i,t\rangle} \lambda t . [t > \textsf{now} \wedge P(t)]$

However, some authors analyze *will* as a modal verb (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. As Cable (2008) points out (building on observations from Otto Jespersen's seminal grammar of English), evidence for this idea comes from the fact that *will* has a past tense variant, *would*:

(66) (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:

(67) WOLL $\rightsquigarrow \lambda P_{\langle i,t\rangle} . \lambda t . [\exists t' . t < t' \wedge P(t')]$

This element can combine with both present and past morphology. Thus the 'future' is not a tense in English. A similar claim has been made for St'át'imcets (Salish) by Matthewson (2006). The sentence *I will mail the letter* will have the representation in Figure 12.2, ignoring the possibility that it may have perfective aspect.

Exercise 7. Assume that *I will mail the letter* involves perfective

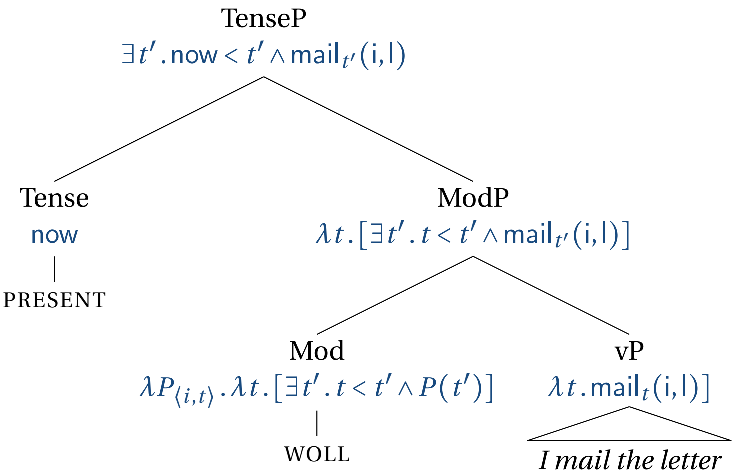


Figure 12.2: Derivation for *I will mail the letter*, ignoring perfective aspect.

aspect, and give a derivation of the truth conditions for that sentence that incorporates this assumption.

Exercise 8. Give a derivation tree for *At 5pm, I would mail the letter.*

12.2.2.5 Perfect

In the spirit of Reichenbach, we adopt the following lexical entry for the English perfect:

(68) PERFECT $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\exists t'.t' < t \wedge P(t')$
'Takes a predicate of times P, and returns a predicate of times that is true of a time t if $\underline{t \text{ follows } t'}$ at which P is true.'

This view is somewhat of an oversimplification; there are a number of different uses for the English perfect, including:

(69) 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.

Exercise 9. Give a derivation tree for *At 5pm, I had mailed the letter.*

12.3 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 view of viewpoint aspect as a way of signalling inclusion relations between temporal intervals, the future as a kind of aspectual operator, and the English perfect as a device for shifting the event time prior to the reference time.

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:

(70) 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)

(71) John said he would buy a fish that **was** still alive. (Ogihara, 1989)

(72) 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 & Gronn (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.

13 Intensional semantics

13.1 Necessity and possibility

13.1.1 Necessary vs. contingent

This chapter deals with modality. To a first approximation, modality has to do with ways of being true. Two true sentences can be true in different ways. For instance, both of the following sentences are true.

(1) 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. (1a) expresses something which might well have turned out to be false. (1b) 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. (As usual, by PROPOSITION, we mean 'something that can be true or false'. 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 (1a) is contingent and (1b) is necessary, they give rise to claims of different truth status when they are embedded under *necessarily*:

(2) 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 (1a) is possible while the negation of (1b) is not:

(3) 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. The tools we develop in this chapter are heavily influenced by Montague's (1974b) Intensional Logic (IL), but ultimately we will settle on a manner of representing intensional phenomena that adheres to Gallin's (1975) Ty2, a logic that includes a type s for possible worlds.

Exercise 1. Give an example of two *contingent* sentences of English that have the same truth value relative to the actual world 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.

13.1.2 Modality and inference

The logic of necessity and possibility mirrors the logic of universal and existential quantification. The negation of a necessity statement like (4a) is equivalent to a possibility statement about a negation as in (4b).

(4) a. It's **not necessary** for you to take out the trash.
    b. It's **possible** for you **not** to take out the trash.

Similarly, to the negate the possibility of a negation is to state the necessity of what is negated:

(5) a. It's **not possible** for me **not** to pay the fine.
    b. It's **necessary** for me to pay the fine.

These patterns can be understood under a treatment of these operators in a logic where necessity involves universal quantification and possibility involves existential quantification.

In MODAL LOGIC, there is a formal representation language in which it is possible to make statements about necessity and possibility. The most widespread semantics for modal logic, 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.

Modal logic does not involve explicit quantification over possible worlds in the representation language; instead, two operators are used, $\Box$ 'box', expressing necessity, and $\Diamond$ 'diamond', expressing possibility. These operators are added to propositional logic or to predicate logic, yielding propositional modal logic or propositional predicate logic respectively.

As a first approximation, $\Box\phi$ is true just in case ϕ is true in every possible world; and $\Diamond\phi$ is true just in case ϕ is true in some possible world. Under this semantics, $\Box$ and $\Diamond$ are, essentially, quantifiers over hidden variables. 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 (5a) and (5b). In this way, $\Box$ and $\Diamond$ are *duals* of each other, just like $\forall$ and $\exists$ are.

In fact, it is possible to simulate modal logic in predicate logic by bringing out the hidden variables 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 $\mathsf{annapayfine}$ be a predicate of worlds that holds of a world w if Anna pays a fine in w. Then when Anna utters (5b) (*It's necessary for me to pay a fine*), the content of her utterance can be captured using the following kind of formula, with universal quantification over possible worlds:

(6) $\forall w.\mathsf{annapayfine}(w)$

(Compare the representation in modal logic: $\Box\mathsf{annapayfine}$.) Likewise, the equivalent sentence invoking possibility ((5a); *It's not* *possible for me to not pay a fine*) can be represented using existential quantification:

(7) $\neg\exists w.\neg\mathsf{annapayfine}(w)$

(Compare the modal logic representation: $\neg\Diamond\neg\mathsf{annapayfine}$.) We can leverage what we know about quantification in predicate logic to understand why (6) and (7) are equivalent.

For simple cases like the equivalence between (5a) and (5b), modal logic and its simulation in predicate logic amount to the same thing. But we will see later that there are things one can express with explicit quantification over worlds that are impossible to express in modal logic. Another deficiency of the simple system of modal logic we have presented so far is that it glosses over distinctions among modal 'flavors', which we discuss next.

13.1.3 Modals: Strength and flavor

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:

(8) 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 *possi-*

ble. 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. Examples (5a) *It's not possible for me to not pay the fine* and (5b) *It's necessary for me to pay the fine* express necessity in relation to a set of rules established by a society (obligation, in other words), so they involve deontic modality. BOULETIC modality has to do with the desires of a given agent; the verb *want* expresses bouletic modality.

In English, many modal expressions are ambiguous between different modal flavors, and are disambiguated by tense, context, and other factors.

(9) 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 *ob-*

jective: 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 deontic modals relate to each other as duals; for example *You must not* ϕ is equivalent to *It is not allowed to* ϕ.

Later in the chapter, we will capture the logical interaction between strong and weak modals while allowing for the nuance of modal flavor by letting different flavors of modality involve quantification over different sets of possible worlds. For metaphysical modals, these are all the possible worlds that are metaphysically possible; for epistemic modals, they are all the possible worlds compatible with the speaker's (or the relevant agent's) knowledge; for deontic modals, these are all the possible worlds compatible with the speaker's (or relevant agent's) obligations.[1]

13.2 Intension vs. extension

In building a compositional system for handling modality, it is useful to distinguish between two types of denotation that an expression can have: intension and extension. Modal words like *necessarily*, for example, are sensitive to the intensions of the sentences that they combine with. Let us explain what what we mean by that.

[1] This insight is developed in work by Angelika Kratzer (e.g. Kratzer 1981), who distinguishes between a 'modal base' (a set of possible worlds) and an 'ordering source' (a set of propositions that can be used to rank worlds); see for example Kratzer (1981).

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.

This example was used by Quine (1951) in a discussion of the difference between INTENSION and EXTENSION. 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. Given the actual world, the intension of *creature with a heart* returns the same set that the intension of *creature with a kidney* does. But there are other worlds for which the sets returned are different. So they are not COINTENSIONAL; they do not have the same intension. Another way of putting it is that they are not INTENSIONALLY EQUIVALENT.

Referential expressions have intensions and extensions too, and we can find examples of referential expressions that are coextensional but not cointensional. For example, the definite description (10a) happens to be coextensional with the name (10b):

(10) a. the first person to serve on both *Harvard Law Review* and *Columbia Law Review*
b. Ruth Bader Ginsburg

In the actual world, these two expressions 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 (10b) still refers to RBG (she is still the same *person*), but the definite description in (10a) has a different referent.

We can make sense of this intuition by identifying the extension of each name or definite description with a particular individual, and by identifying the intension of each name or definite description with a (possibly partial) function from possible worlds to individuals. Such a function is called an INDIVIDUAL CONCEPT. We treat the intension of (10a) as a partial function which maps each world to whichever woman, if any, was the first to serve on both law reviews *in that world*. By contrast, we treat the intension of (10b) as mapping every possible world (or at least every possible world in which RBG exists) to RBG. This kind of function is called a RIGID DESIGNATOR; Kripke (1979) famously argued that proper names denote rigid designators. On this view, the two noun phrases in (10) have different intensions, although they are coextensional.

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, these (1a) and (1b) are coextensional, because they are both true.

Clearly, the extension of a sentence cannot be its meaning; otherwise (1a) and (1b) 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. Assume that propositions are the kinds of things that one believes, knows, doubts, etc. Since someone might doubt (1a) without doubting (1b), or might believe (1b) without believing (1a), 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 $\mathsf{T}$and others to F. Since (1a) is contingent and (1b) 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.

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.

13.3 Opacity puzzles

13.3.1 Substitutability and its failures

With the distinction between intension and extension in hand, let us now present a puzzle that was observed by Gottlob Frege. Frege observed that the logic of natural language does not always adhere to the following principle:

(11) **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 (recall the discussion in Chapter 1).

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:

(12) 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 (3a) and (3b), repeated here:

(13) 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 (13a) 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 (13a), 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, (13b) is clearly false. (Here, we are focusing on the reading of (13b) that is similar to *RBG might have been someone else* or *RBG might have been distinct from herself.* To the extent that (13b) 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 (13b) 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.)

As with modals, we find violations of the principle of substitutability of coextensionals with verbs expressing attitudes towards propositions (i.e. PROPOSITIONAL ATTITUDE VERBS) such as *believe*, *know* and *want*. We find violations of substitutability of coextensionals here as well. For example, (14) does not imply (15).

(14) Mary believes that RBG is RBG.

(15) Mary believes that RBG is the first person to serve on *Harvard Law Review* and *Columbia Law Review*.

Environments in which the principle of substitutability of coextensionals fails are called OPAQUE. As we have seen, modals and propositional attitude verbs give rise to 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 *extension* 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. The solution will build on the idea that intensional operators combine with the intensions, rather than the extensions, of their complements. For example, belief is a relation between the subject of *believe* and the *proposition* (rather than the truth value) of the sentential clause embedded by *believe*.

13.3.2 Veridicality

Another valid argument involving modal operators involves the inference from necessary truth to simple truth:

(16) 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:

(17) a. Possibly, this bottle is made of plastic.
b. $\not\therefore$ 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.

A similar distinction can be found within propositional attitude verbs. For example, *know*, *notice*, and *see* are veridical, but *believe* and *doubt* are not. Thus, (18a) does not entail (18b):

(18) a. Mary believes that Fred is in Paris.
b. $\not\therefore$ Fred is in Paris.

Similarly, (19a) does not entail (19b).

(19) a. Mary believes that a unicorn is eating her parsnips.
b. $\not\therefore$ A unicorn is eating Mary's parsnips.

A theory of modality should capture the fact that *necessarily* is veridical, but *believe* is not.

13.3.3 Existential import

In fact, (19a) does not even entail that there are unicorns; all it entails is that Mary believes that there are unicorns. In other words, (19a) lacks EXISTENTIAL IMPORT with respect to the indefinite *a unicorn.*

Propositional attitudes can also be expressed by transitive verbs that take a noun phrase direct object, and in such cases we observe the same effect. Thus, while (20a) implies that there is at least one sloop (a type of sailboat), (20b) need not do so:

(20) a. Andrea sees a sloop.
b. Andrea wants a sloop.

As Quine (1956) puts it, example (20b) 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. Thus a representation of the following kind would not do:

Borrowing Quine's metaphor, verbs like *believe, want,* and *look for* 'seal off' the complement clause. As a consequence, the existential import of the complement clause is not inherited by the sentence as a whole. Lack of existential import under such verbs is among the phenomena we wish to capture in this chapter.

13.3.4 *De dicto* vs. *de re*

Although (20b) 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').[2] On the *de re* reading, Andrea has a de-

[2] A similar distinction may have been anticipated by Aristotle and is discussed by the medieval logician Abelard, whom we encountered in Chapter 4. The terms themselves appear for the first time about one century later in the writings of Saint Thomas Aquinas, the philosopher and theologian.

sire 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.

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.

(21) 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 would have been 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.

Quine (1956) illustrated the *de dicto* / *de re* ambiguity with examples like the following:

(22) a. Ralph believes that someone is a spy.
b. Ralph believes that the man in the brown hat is a spy.

Consider first example (22a). On the *de re* reading, Ralph has a belief about a particular object/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. The *de dicto* interpretation does not entail that there are, in fact, any spies; it only entails that Ralph believes there is one. For example (22b), 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.

Consider another example:

(23) 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.[3]

Exercise 4. 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

[3] 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*).

> 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.

In the following section, we will build up some tools that will allow us to capture the *de dicto* vs. *de re* ambiguity, to explain the lack of existential import with opaque verbs, and to explain why coextensional expressions cannot always be substituted for each other. We will start with a discussion of intensional models, which include a set of possible worlds. We will then present Gallin's Ty2, a logical representation language with two types other than t (hence the name), one for individuals as before (e) and one for possible worlds (s).

13.4 Representing intensionality

We now modify our system from previous chapters so that it provides access to the propositions expressed by sentences. More generally, the system we develop in this section provides access to the intensions of expressions along with their extensions.

13.4.1 Intensional models

Let us return to this example:

(24) Mariel wants to dance with a drummer.

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.' Before we develop a logical representation language that we can use to capture these two readings, let us first consider what it is that we want to represent. 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. So it is important to be able to evaluate the truth of a claim relative to various different worlds. The WORLD OF EVALUATION is the world relative to which we are evaluating the truth of a sentence. If one is interested in truth relative to the actual world, then the world of evaluation should be the actual world.

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 (24) 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.

To provide an ontological foundation for this idea, it will be useful to incorporate possible worlds into our models. So rather than assuming that a model M is just a pair $\langle D, I \rangle$, where D is a set of individuals and I is an interpretation function assigning denotations to each of the non-logical constants, as we have been doing, let us assume that a model M further specifies a set of worlds W. A so-called KRIPKE MODEL M is a triple $\langle D, I, W \rangle$, specifying a domain of individuals, an interpretation function, and a set of possible worlds.

For purposes of illustration, let us consider a model where in some of the worlds in W, Mariel dances with a drummer. Let us assume that there are only four worlds, w_0, w_1, w_2, and w_3. Assume that in w_1 and w_2, Mariel dances only with Ruth. In w_3, she dances only with Leo. In w_0, she dances with nobody. This information, along with who is a drummer in each world, is summarized in Table 13.1. Notice that in w_2, Mariel dances only with Ruth, and since Ruth is not a drummer in that world, it is not true in w_2 that Mariel dances with a drummer. In w_3, Mariel dances

	w_0	w_1	w_2	w_3
drummers	Ruth	Ruth	Leo	Leo
Mariel dances with	nobody	Ruth	Ruth	Leo
Mariel desires	$\{w_1, w_3\}$	$\{w_1, w_2\}$	$\{w_1, w_2\}$	$\{w_0\}$
(24) *de dicto*	T	F	F	F
(24) *de re*	F	T	F	F

Table 13.1: Above midline: Partial specification of a Kripke model. Below midline: truth status of two readings of (24) *Mariel wants to dance with a drummer* in each world. *Mariel dances with a drummer* is true in w_1 and w_3.

only with Leo, and since Leo is a drummer in w_3, it's true in w_3 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 w_3, she doesn't want to dance with anybody, even though she happens to be dancing with Leo. In other words, in w_3, the only world among her desire worlds (the worlds that satisfy her desires) is w_0. Mariel's desire worlds in each world are likewise specified as in Table 13.1. 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 the information we need in order to evaluate the truth or falsity of (24) (*Mariel wants to dance with a drummer*) on various readings at various worlds.

In w_3, the sentence is false on any reading. In w_3, Mariel just doesn't want to dance with anybody, drummer or not.

Notice that in w_0, Mariel's desire worlds are w_1 (where she dances with Ruth, a drummer in that world) and w_3 (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 w_0, the *de dicto* reading of (24) is true.

In w_1, Mariel's desire worlds are w_1 and w_2. In both of these worlds, she dances with Ruth, who happens to be a drummer in w_1, though not in w_2. In w_1, then, Mariel wants *de re* to dance with someone, namely Ruth; and Ruth is a drummer. So the *de re* reading of (24) is true in w_1. This is so even though Ruth is not a drummer in w_2 — but the *de dicto* reading of (24) is false in w_1 for that reason.

Finally, in w_2, Mariel's desire worlds are again w_1 and w_2. She just wants to dance with Ruth. But Ruth is not a drummer in w_2. So although in w_2 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 w_2, an is its *de dicto* reading.

13.4.2 Representing worlds explicitly

We now present a theory of modality in the style of Gallin (1975), which, in contrast to modal logic and Montague's Intensional Logic, brings out the hidden world variables out into the open, just as we have seen in the simulation of modal logic in predicate logic in Section 13.1.2. Gallin's system is called Ty2, in reference to the fact that along with the basic type t, there are two other basic types: e for individuals and s for worlds. The fact that s is a basic type means that there are expressions of the language that denote possible worlds. These may in principle be either constants, picking out particular worlds, or variables. For variables of type s, we will use indexed strings of the form w_i, where i is an integer, such as w_0, w_1, w_2, etc. We use w as an abbreviation for w_0 and w' as an abbreviation for w_1, etc.

By exposing world variables and making them accessible to quantifiers, Ty_2 provides the means to account for contrasts that are beyond the reach of the boxes and diamonds of modal logic and Intensional Logic. The following is based on a classical example due to Cresswell (2012):

(25) 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 Intensional Logic:

(26) $\Diamond \forall x[\mathsf{rich}(x) \to \mathsf{poor}(x)]$

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 in IL using boxes and diamonds and hats. In particular, this won't work:

(27) $\forall x[\mathsf{rich}(x) \to \Diamond \mathsf{poor}(x)]$

This says that for every rich person, there is a possibility that that person could have been poor. It could be different possibilities for different rich people. Or 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.

The relevant reading can be captured by the following Ty_2 formula, where w is the world of evaluation, and w' is the counterfactual world the sentence is talking about:

(28) $\exists w'.\forall x.[\mathsf{rich}_w(x) \to \mathsf{poor}_{w'}(x)]$

Hence it seems that natural language requires explicit use of world variables, as in Ty2.

13.4.3 Definitions for Ty2

13.4.3.1 Syntax of Ty2

The syntax of Ty2 is exactly the same as the syntax for L_λ, with the exception that there are now *two* basic types along with t, namely e and s (hence the name, 'Ty2'). That means in particular that there are expressions of type s, denoting possible worlds. As usual, if σ and τ are types, then so is $\langle\sigma,\tau\rangle$.

We take from L_λ the rules for forming basic expressions: For every type τ, there is:

- a possibly empty set of constants Con_τ
- an infinite set of variables Var_τ, each bearing a natural number as an index, one for each natural number. (The index 0 can be suppressed, so x is an abbreviation of x_0. Primes may also be used in place of numbers, so x' abbreviates x_1, and x'' abbreviates x_2, etc. Abbreviated and non-abbreviated forms should not occur in the same formula, to avoid confusion.)

As before, we use the following typing conventions:

- variables of the form x_i, y_i or z_i; where i is an integer, are variables of type e;
- variables of the form P_i or Q_i are of type $\langle e,t\rangle$;
- variables of the form R_i are of type $\langle e,\langle e,t\rangle\rangle$.

In addition:

- variables of the form w_i are of type s;
- variables of the form p_i are of type $\langle s,t\rangle$;
- variables of the form $\mathcal{P}_i$ are of type $\langle s,\langle e,t\rangle\rangle$.

Outside of these conventions, we sometimes indicate the type of a variable by means of an additional subscript.

To form complex expressions, Ty2 includes the rule of Application, forming type τ expressions of the form $[\alpha(\beta)]$ where α has type $\langle\sigma,\tau\rangle$ and β has type σ. When β is of type s, we write α_β as a shorthand for $\alpha(\beta)$. Identity statements can be formed as usual: If α and β are expressions of the same type, then $\alpha = \beta$ is a formula (an expression of type t). The connectives of propositional logic apply to expressions of type t in the usual way, and $\forall$ and $\exists$ also create formulas in the usual way, quantifying over variables of any type. Lambda abstraction is also defined as usual: 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 $\langle\sigma,\tau\rangle$.

13.4.3.2 Semantics of Ty2

The semantic value of a well-formed expression α is defined relative to a Kripke model M, which specifies a set W of possible worlds along with a set of individuals D. Let $\mathcal{T}$ be the set of types (e for individuals, t for truth values, s for possible worlds, $\langle e,t\rangle$ for functions from individuals to truth values, etc.). For each type $\tau \in \mathcal{T}$, the model determines a corresponding domain D_τ. Following our usual manner, we define the 'standard frame based on D and W' as an indexed family of sets $(D_\tau)_{\tau\in\mathcal{T}}$, where:

- $D_e = D$
- $D_s = W$
- $D_t = \{\mathsf{T},\mathsf{F}\}$
- 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 Ty2 based on D and W, then, is a pair $\langle(D_\tau)_{\tau\in\mathcal{T}}, I\rangle$, where $(D_\tau)_\tau$ is a standard frame based on D and W.

The interpretation function I has all of the non-logical constants in its domain, and associates a denotation in D_τ to every non-logical constant of type τ, just as in L_λ. If α is a non-logical constant, then $[\![\alpha]\!]^{M,g} = I(\alpha)$. And as usual, if α is a variable, then $[\![\alpha]\!]^{M,g} = g(\alpha)$. The semantic rules for the logical connectives and the variable binders are exactly the same as in L_λ as well. No surprise, then, that equality is defined just as in L_λ too: $[\![\alpha = \beta]\!]^{M,g} = \mathsf{T}$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$.

The idea that an individual might exist in one world but not in another can be represented using a world-dependent existence predicate of type $\langle s, \langle e, t\rangle\rangle$. For instance, the following expression:

(29) $\quad \mathsf{exists}_w(x)$

can be used to express the idea that 'x' exists in w. Doing so commits us to a so-called POSSIBILIST view on existence, where there are things (in the model) which do not exist (in whichever world we happen to be in) but could exist. On an ACTUALIST view, there is nothing that could exist but doesn't actually exist.[4]

13.4.4 Translating from English to Ty2

Now let consider how to translate a few selected expressions of a natural language, namely English, into Ty2. The lexicon of this fragment will be set up in such a way as to specify, for each lexical item, a translation into Ty2 that represents its extension at a given world of evaluation.[5] Following Gallin (1975), English expressions whose extension depends on an evaluation world will be trans-

[4]For more on possibilism vs. actualism, see the Stanford Encyclopedia of Philosophy entry on 'Actualism'.

[5]In this way, we follow in the footsteps of Groenendijk & Stokhof (1982) and Clifford (1990), although the latter treats logical words like *every* differently than we do here, and has a slightly different system of semantic composition. Other systems that use explicit world variables, like the one used in Romero (2008), involve a translation relation that associates a natural language expression with its intension instead of its extension at a given world.

lated into Ty2 using expressions that contain a free variable for the evaluation world. The world of evaluation is represented by the variable w_0, also notated as $\dot{w}$. This is a variable rather than a constant, so a given Ty2 translation α that makes reference to $\dot{w}$ may have a different semantic value $[\![\alpha]\!]^{M,g}$ depending on what the assignment function g assigns to $\dot{w}$. The intension of a natural language expression can then be represented simply by prepending $\lambda\dot{w}$ to the representation for its extension.

13.4.4.1 Lexicon

Common nouns. A common noun like *drummer* or *unicorn* has a translation of type $\langle e,t\rangle$, with the world slot of the corresponding $\langle s,et\rangle$ predicate saturated by the distinguished evaluation world variable $\dot{w}$:

(30) *drummer* $\rightsquigarrow \lambda x\,.\,\mathsf{drummer}_{\dot{w}}(x)$ $\langle e,t\rangle$

(31) *unicorn* $\rightsquigarrow \lambda x\,.\,\mathsf{unicorn}_{\dot{w}}(x)$ $\langle e,t\rangle$

Names. A name like *Miss America* can in principle have different values depending on the world of evaluation (and is temporally variable too, but let us set that aside). Let us therefore represent its meaning with a constant ma of type $\langle s,e\rangle$, which yields a particular individual given a world. Since $\mathsf{ma}_{\dot{w}}$ is an expression of type e, it denotes an individual.

(32) *Miss America* $\rightsquigarrow \mathsf{ma}_{\dot{w}}$ e

For ordinary proper names like *Leo* and *Mariel*, we may wish to treat them as rigid designators as Kripke (1980) proposed, picking out an individual directly, regardless of the world of evaluation. One way of doing this is to translate a name like *Mariel* with a constant of type $\langle s,e\rangle$, just like *Miss America*. Then our lexical entry for *Mariel* might look like this:

(33) *Mariel* $\rightsquigarrow \mathsf{m}_{\dot{w}}$ e
(where m is a constant of type $\langle s, e\rangle$)

We may assume further that m has as its intension a constant function on worlds, giving the same output regardless of the input.

A more syntactic way to enforce the idea that *Mariel* is a rigid designator is to specify its extension with a constant of type e:

(34) *Mariel* $\rightsquigarrow \mathsf{m}$ e
(where m is a constant of type e)

Since the latter approach involves writing fewer symbols, we will adopt it, even though it does take us one tiptoe step outside the strict boundaries of Gallin's translation from IL to Ty2.

Determiners. In our move to Ty2 we do not change the translations of logical words such as quantificational determiners. For example, *every* has a translation of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$, as usual:

(35) *every* $\rightsquigarrow \lambda P_{et} \lambda Q_{et} \forall x[P(x) \rightarrow Q(x)]$ $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$

Its denotation is *not* sensitive to the evaluation world. This reflects the view that it is a logical word of English, rather than a content word.

Modals. To a first approximation, an intensional adverb like *necessarily* can be represented using universal quantification over worlds, and *possibly* using existential quantification.

(36) *necessarily* $\rightsquigarrow \lambda p_{st} . \forall w' . p(w')$ $\langle st, t\rangle$

(37) *possibly* $\rightsquigarrow \lambda p_{st} . \exists w' . p(w')$ $\langle st, t\rangle$

But this treatment ignores modal flavor.

To capture modal flavor, one can restrict the set of worlds that are being quantified over. One mechanism for restricting the set of worlds that are quantified over is an ACCESSIBILITY RELATION among possible worlds, an idea from modal logic. An accessibility relation specifies for each world, which other worlds are accessible to it. An accessiblity relation is a binary relation among worlds, a set of pairs $\langle w, w' \rangle$ such that w' is accessible from w. In our earlier discussion of modal logic, we omitted the distinction between accessible worlds and inaccessible worlds, but the $\Box$ and $\Diamond$ operators are usually defined so as to quantify only over accessible worlds:

- $\Box\phi$ is true at a given world w if and only if for all worlds w' accessible to w, ϕ is true at w'.
- $\Diamond\phi$ is true at a given world w if and only if there is a world w' accessible to w such that ϕ is true at w'.

In the simplest case, there is only one accessiblity relation 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 algebraic properties of the accessibility relation.

To capture modal flavor, one strategy is to adopt multiple different accessiblity relations. Thus a world w' might 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.

Let us use $\mathsf{deon.acc}_w(w')$ to represent the idea that w' is deontically accessible from w. To say that w' is deontically accessible from w is to say that w' is a world that is compatible with the rules that are in place in w. Let's assume further that *necessarily* is polysemous, with different lexical entries for each flavor. Then the deontic sense of *necessarily* can be written as follows:

(38) $\quad$ *necessarily*$_{\text{DEON}} \rightsquigarrow \lambda p_{st} \,.\, \forall w' [\mathsf{deon.acc}_{\dot{w}}(w') \rightarrow p(w')]$

Epistemic and metaphysical necessity can be modelled in terms of an other accessibility relations:

(39) *necessarily*$_{\text{EPIST}} \rightsquigarrow \lambda p_{st}\,.\,\forall w'[\mathsf{epist.acc}_{\dot{w}}(w') \rightarrow p(w')]$

(40) *necessarily*$_{\text{META}} \rightsquigarrow \lambda p_{st}\,.\,\forall w'[\mathsf{meta.acc}_{\dot{w}}(w') \rightarrow p(w')]$

The corresponding senses of *possibly* would be analogous, using existential instead of universal quantification, for example:

(41) *possibly*$_{\text{META}} \rightsquigarrow \lambda p_{st}\,.\,\exists w'[\mathsf{meta.acc}_{\dot{w}}(w') \rightarrow p(w')]$

We will abbreviate $\exists w'[\mathsf{meta.acc}_{w}(w') \rightarrow p(w')]$ as $\mathsf{poss.meta}_{w}(p)$. Likewise, $\forall w'[\mathsf{meta.acc}_{w}(w') \rightarrow p(w')]$ is short for $\mathsf{neg.meta}_{w}(p)$. *Mutatis mutandis* for the other flavors.

One fact that this treatment leaves out is the context-sensitive nature of modals. The modal verb *have to* is unambiguously deontic in flavor, but still, the relevant set of rules in question may differ from context to context. In some cases, the speaker might be talking about what is legal according to the rules of the local country; in others, it might be rules of the house that are in question. To accommodate this fact, we could represent the meaning of a sentence involving *have to* using a free variable that can get its value from context. This variable will specify, for a given world, which (other) possible worlds conform to the relevant set of rules in force. It can therefore be modelled as a relation between possible worlds. We'll use the variable $\mathcal{R}$, of type $\langle s, \langle s, t\rangle\rangle$, to represent it:

(42) *have to* $\rightsquigarrow \lambda p\,.\,\forall w'[\mathcal{R}_{\dot{w}}(w') \rightarrow p(w')]$

The denotation of $\mathcal{R}$ relative to a model M and an assignment function g depends directly on the value that g assigns to $\mathcal{R}$. This lexical entry for *have to* in (42) doesn't place any constraints at all on $\mathcal{R}$, but a presupposition that $\mathcal{R}$ is of a deontic nature could be added in a system using a version of Ty2 that supports the ∂ operator.

Propositional attitude verbs. Like modals, intensional verbs like *believe* and *want* expect intensional inputs. In particular, these two verbs expect type $\langle s, t\rangle$. The lexical entry for *believes* makes reference to a non-logical constant dox, a function that specifies for a given world and a given agent which other possible worlds are compatible with the agent's beliefs (cf. Hintikka 1969). The function is called dox after the word 'doxastic', a Greek word meaning 'belief'; belief is a doxastic attitude. If a possible world is compatible with an agent's beliefs (at a given world of evaluation), the say that this world is DOXASTICALLY ACCESSIBLE to the agent (at the world of evaluation). To believe a proposition p, then, is for that proposition to hold at all of the agent's doxastically accessible worlds.

(43) *believes* $\rightsquigarrow \lambda p_{st} \lambda x \,.\, \forall w' [\mathsf{dox}_{\dot{w}}(x, w') \rightarrow p(w')]$

We abbreviate $\forall w' [\mathsf{dox}_u(\alpha, w') \rightarrow p(w')]$ as $\mathsf{bel}_u(\alpha, p)$.

Similarly, to want a proposition p is for that proposition to hold at all of the agent's desire worlds, designated with boul for 'bouletic':

(44) *wants* $\rightsquigarrow \lambda p_{st} \lambda x \,.\, \forall w' [\mathsf{boul}_{\dot{w}}(x, w') \rightarrow p(w')]$

We abbreviate $\forall w' [\mathsf{boul}_u(\alpha, w') \rightarrow p(w')]$ as $\mathsf{want}_u(\alpha, p)$.

This concludes the exposition of our toy lexicon.

Exercise 5. *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* $\rightsquigarrow \lambda p_{st} \lambda x \,.\, \forall w' [\mathsf{dox}_{\dot{w}}(x, w') \leftrightarrow p(w')]$

(ii) *believes* $\rightsquigarrow \lambda p_{st} \lambda x \,.\, \exists w' [\mathsf{dox}_{\dot{w}}(x, w') \wedge p(w')]$

Both are very wrong. Explain why these do not adequately capture the meaning of *believe*.

13.4.4.2 Composition rules

Let us turn now to composition rules. The Function Application rule that we use in this system is exactly as before:

Composition Rule 7. 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

$$\gamma \rightsquigarrow \alpha'(\beta')$$

Hence for example:

(45) *every boy* $\rightsquigarrow \lambda Q_{et} \forall x[\mathsf{boy}_{\dot{w}}(x) \rightarrow Q(x)]$

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) proposed 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 8. 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
$$\gamma \rightsquigarrow \alpha'(\lambda \dot{w} \,.\, \beta')$$

We also continue to adopt the Pronouns and Traces rule, according to which a pronoun gets translated as a variable with the corresponding numerical index (here we are glossing over gender for expository purposes):

(46) *she*$_i \rightsquigarrow x_i$

And as usual, we will use the transformation rule of Quantifier Raising and the composition rule of Predicate Abstraction. The composition rule of Predicate Modification comes along as well.

With these rules, we derive the representation in (48) for (47).

(47) She believes that Leo is a drummer.

(48) $\mathsf{bel}_{\dot{w}}(x_i, \lambda \dot{w}.\mathsf{drummer}_{\dot{w}}(\mathsf{l}))$

x_i
She$_i$

$\lambda x.\mathsf{bel}_{\dot{w}}(x, \lambda \dot{w}.\mathsf{drummer}_{\dot{w}}(\mathsf{l}))$

$\lambda p_{st} \lambda x.\mathsf{bel}_{\dot{w}}(x, p)$
believes

$\mathsf{drummer}_{\dot{w}}(\mathsf{l})$

(that)

l
Leo

$\lambda x.\mathsf{drummer}_{\dot{w}}(\mathsf{x})$
is a drummer

Notice that the translation we obtain for this example involves both a free and a bound use of the designated variable $\dot{w}$. Due to the fact that bound variables can be re-lettered to produce an equivalent expression via the alpha-conversion rule, the translation we obtain at the top of the tree in (48) is equivalent to:

(49) $\mathsf{bel}_{\dot{w}}(\mathsf{m}, \lambda w'.\mathsf{drummer}_{w'}(\mathsf{l}))$

(The free instance of $\dot{w}$ cannot be replaced by another world variable to produce an equivalent meaning, though.)

13.5 Applications

13.5.1 Substitutability and its failures

13.5.1.1 Identity statements

The tools we have developed allow us to account can be given for the contrast between:

(50) a. RBG might not have been the first person to serve on *Harvard Law Review* and *Columbia Law Review*
b. RBG might not have been RBG

Let us assume that modal verb *might* here is a synonym of *possibly* in its metaphysical sense. This gives the following translations for the two sentences (we use $\mathsf{firstwoman}$ to denote the first person to serve on *Harvard Law Review* and *Columbia Law Review*):

(51) a. $\mathsf{poss.meta}_{\dot{w}}(\lambda w'.\neg[\mathsf{rbg} = \iota x[\mathsf{firstwoman}_{w'}(x)]])$
b. $\mathsf{poss.meta}_{\dot{w}}(\lambda w'.\neg[\mathsf{rbg} = \mathsf{rbg}])$

In the first case, the sentence expresses metaphysical possibility with respect to a proposition whose truth depends on who is the first woman to serve on both law reviews (and in particular whether that individual happens to be Ruth Bader Ginzburg). In the second case, the sentence expresses the possibility of a proposition that is clearly necessarily false. Hence the contrast.

Exercise 6. Give a compositional derivation for (50a) and (50b) using the tools we have developed in this chapter and a Fregean treatment of the definite article. You may assume that *first person to serve on the HLR and the CLR* is a single word. At each node, specify not only the translation into Ty2, but also the semantic type and the composition rule that was used to derive it. Assume an LF on which *might* combines with a complete sentence.

We also have tools for explaining the invalidity of the following argument:

(52) Camille is Miss America.
Necessarily, Camille is Camille.
∴ Necessarily, Camille is Miss America.

Translating the argument into Ty2 according to the rules we have laid out, and treating of *is* as equality, gives the following:

(53) $\mathsf{c} = \mathsf{ma}_{\dot{w}}$
$\mathsf{nec.meta}_{\dot{w}}(\lambda w.\mathsf{c} = \mathsf{c})$

$$\therefore \mathsf{nec.meta}_{\dot{w}}(\lambda w\,.\,\mathsf{c} = \mathsf{ma}_{w})$$

The use of explicit world variables in Ty2 helps to bring out where the argument goes wrong. The first premise is a claim about a particular world of evaluation, which can well be true but has no bearing on who is Miss America in other worlds. The second premise is a necessity claim that is obviously true because it concerns an identity that doesn't even depend on a particular world of evaluation. The conclusion, on the other hand, expresses a necessity claim about a proposition whose truth can in principle vary from world to world. The truth of the proposition said to be necessary depends on who Miss America is in the world, and that may or may not be Camille.

13.5.1.2 Frege's Puzzle

In the foregoing, we assumed that proper names like *Ruth Bader Ginzburg* and *Camille* 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.

13.5.2 Existential import

Recall the distinction between *see* and *want* discussed above in connection with examples (20a) *Andrea sees a sloop* and (20b) *Andrea wants a sloop*. The former implies the existence of sloops, while the latter does not. In his seminal work on intensional semantics, Montague discussed a similar contrast between *find*, an extensional verb, and *seek*, an intensional verb:

(54) a. Andrea found a unicorn.
b. Andrea sought a unicorn.

Example (54a) implies that there are unicorns (54b), on one of its readings, does not; to borrow Quine's 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 $\langle e, \langle e, t\rangle\rangle$:

(55) *finds* $\rightsquigarrow \lambda y \lambda x . \mathsf{find}_{\dot{w}}(x, y)$

The translation for *find a unicorn* involves QR of *a unicorn*, giving the LF in (56a) and the Ty2 translation in (56b)

(56) a. LF: [a unicorn] λ_1 [Andrea found t_1]
b. $\exists y[\mathsf{unicorn}(y) \wedge \mathsf{find}_{\dot{w}}(\mathsf{a}, y)]$

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 $\langle s, et\rangle$.

(57) *seeks* $\rightsquigarrow \lambda \mathcal{P}_{\langle s,et\rangle} \lambda x . \mathsf{seek}_{\dot{w}}(x, \mathcal{P})$

Given that *a unicorn* is an expression of type $\langle e, t\rangle$, it can combine with *seek* via Intensional Function Application like so:

(58)

$\mathsf{seek}_{\dot{w}}(\mathsf{a}, \lambda \dot{w} \lambda x . \mathsf{unicorn}_{\dot{w}}(x))$

- a
  Andrea
- $\lambda x . \mathsf{seek}_{\dot{w}}(x, \lambda \dot{w} . \lambda x . \mathsf{unicorn}_{\dot{w}}(x))$
  - $\lambda \mathcal{P}_{\langle s,et\rangle} \lambda x . \mathsf{seek}_{\dot{w}}(x, \mathcal{P})$
    seeks
  - $\mathsf{unicorn}_{\dot{w}}(\mathsf{x})$
    a unicorn

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.

13.5.3 *De dicto* vs. *de re*

Let us now return to the question of how to represent 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.[6] This gives:

(59) 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*:

(60) 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:

(61) [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 (60) (dw is short for 'dance with'):

[6] 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.

(62) $\lambda x\,.\,\mathsf{want}_{\dot{w}}(x, \lambda\dot{w}\,.\,\exists y[\mathsf{drummer}_{\dot{w}}(y) \wedge \mathsf{dw}_{\dot{w}}(x_1, y)])$

$\lambda p_{st}\lambda x\,.\,\mathsf{want}_{\dot{w}}(x, p)$
wants

$\exists y[\mathsf{drummer}_{\dot{w}}(y) \wedge \mathsf{dw}_{\dot{w}}(x_1, y)]$

[a drummer] λ_2 PRO$_i$ (to) dance with t_2

The translation for the full sentence is as follows:

(63) $\mathsf{want}_{\dot{w}}(x_1, \lambda\dot{w}\,.\,\exists y[\mathsf{drummer}_{\dot{w}}(y) \wedge \mathsf{dw}_{\dot{w}}(x_1, y)])$

If we re-letter the bound instance of $\dot{w}$ to w', then the derived truth conditions can be paraphrased as: 'In the world of evaluation (= $\dot{w}$), 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 (63) 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 $\dot{w}$. If g assigns x_1 to Mariel and $\dot{w}$ to w_0, and according to M, w_0 is as described in Table 13.1, then this formula will be true relative to M and g.

Exercise 7. Which reading of *She wants to dance with a drummer* is represented by (63): *de re* or *de dicto* (or neither)?

Exercise 8. Using an appropriate LF, give a compositional derivation for the other reading of *She wants to dance with a drummer*, specifying which reading it is (*de re* or *de dicto*), and give an example of an assignment function relative to which it would be true given the model described in Table 13.1.

13.6 Summary

In this chapter we have developed a compositional intensional semantics using Ty2 as a representation language. We have illustrated how it can be used to capture basic inference patterns involving strong and weak modals of various flavors, how it can be used to block inferences of existential import, and how it can be used to represent *de dicto* vs. *de re* ambiguities. For a book-length presentation of intensional semantics, we recommend von Fintel & Heim (2011).

Bibliography

Abney, Steven Paul. 1987. *The English noun phrase in its sentential aspect.* Cambridge, MA: MIT dissertation.

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.

Anderson, Jill. 2014. Misreading like a lawyer: Cognitive bias in statutory interpretation. *Harvard Law Review* 1521. 1563–68.

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(2). 159–219. doi: 10.1007/bf00350139.

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.

Black, Max & Peter Thomas Geach. 1961. *Translations from the philosophical writings of Gottlob Frege.* Basil Blackwell 2nd edn.

Bochnak, Ryan. 2016. Past time reference in a language with optional tense. *Linguistics and Philosophy* 39. 247–294.

Bresnan, Joan. 2001. *Lexical-functional syntax.* Malden, MA: Blackwell.

Bruening, Benjamin. 2001. *Syntax at the edge: Cross-clausal phenomena and the syntax of Passamaquoddy*: MIT dissertation.

Bruening, Benjamin. 2008. Quantification in Passamaquoddy. In *Quantification: A cross-linguistic perspective*, 67–103. Emerald Group Publishing.

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. 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. 2015. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy* 38(1). 31–66. doi:10.1007/s10988-014-9162-8.

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 & 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.

Church, Alonso. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic* 5. 56–68.

Clifford, James. 1990. *Formal semantics and pragmatics for natural language querying.* Cambridge: Cambridge University Press.

Comrie, Bernard. 1976. *Aspect.* Cambridge: Cambridge University Press.

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.

Evans, Gareth. 1977. Pronouns, quantifiers, and relative clauses. *Canadian Journal of Philosophy* 7.

Farkas, Donka. 2002. Specificity distinctions. *Journal of Semantics* 19(3). 213–43.

Fauconnier, Gilles. 1975. Pragmatic scales and logical structure. *Linguistic Inquiry* 6(3). 353–375.

von Fintel, Kai. 1994. *Restrictions on quantifier domains*: University of Massachusetts at Amherst dissertation.

von Fintel, Kai. 1999. NPI licensing, Strawson entailment, and context-dependency. *Journal of Semantics* 16. 97–148.

von Fintel, Kai. 2004. Would you believe it? The King of France is back! (presuppositions and truth-value intuitions). In

A. Bezuidenhout & M. Reimer (eds.), *Descriptions and beyond*, 315–342. Oxford: Oxford University Press.

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. 1891. *Function und Begriff.* Jena, Germany: Hermann Pohle. Translated in Black & Geach (1961), 21-41.

Frege, Gottlob. 1892 [reprinted 1948]. Sense and reference. *The Philosophical Review* 57(3). 209–230.

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.

Geach, Peter. 1962. *Reference and generality.* Ithaca, NY: Cornell University Press.

Geurts, Bart & David Beaver. 2011. Discourse representation theory. In Edward Zalta, Uri Nodelman & Colin Allen (eds.), *Stanford encyclopedia of philosophy*, CSLI, Stanford.

Giannakidou, Anastasia. 1999. Affective dependencies. *Linguistics and Philosophy* 22. 367–421.

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, J. & M. Stokhof. 1982. Semantic analysis of wh-complements. *Linguistics and Philosophy* 5(2). 175–233.

Groenendijk, Jeroen, Theo Janssen & Martin Stokhof (eds.). 1984. *Truth, interpretation and information: selected papers from the third amsterdam colloquium*. Dordrecht, Netherlands: Foris.

Groenendijk, Jeroen & Martin Stokhof. 1990a. Dynamic Montague grammar. *Proceedings of the Second Symposion on Logic and Language* 3–48.

Groenendijk, Jeroen & Martin Stokhof. 1990b. Dynamic Montague grammar. In *Proceedings of the second symposium on logic and language*, 3–48. Budapest.

Groenendijk, Jeroen & Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14. 39–100.

Gruber, Jeffrey S. 1965. *Studies in lexical relations*. Cambridge, MA: Massachusetts Institute of Technology dissertation. `http://hdl.handle.net/1721.1/13010`.

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. 1982a. *On the semantics of definite and indefinite noun phrases*: Umass. Amherst dissertation.

Heim, Irene. 1982b. *The semantics of definite and indefinite noun phrases*: U. Mass Amherst dissertation.

Heim, Irene. 1983a. File change semantics and the familiarity theory of definiteness. In Rainer Bäuerle, Christoph Schwarze & Arnim von Stechow (eds.), *Meaning, use, and the interpretation of language*, 164–189. Berlin: Walter de Gruyter.

Heim, Irene. 1983b. 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. 1992. Presupposition projection and the semantics of attitude verbs. *Journal of Semantics* 9. 183–221.

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. 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. 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.

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.

Karttunen, L. 1973a. Presuppositions of compound sentences. *Linguistic inquiry* 4(2). 169–193.

Karttunen, Lauri. 1973b. Presuppositions of compound sentences. *Linguistic Inquiry* 4(2). 169–193.

Karttunen, Lauri. 1976. Discourse referents. In James D. McCawley (ed.), *Notes from the linguistic underground* (Syntax and Semantics 7), 363–385. New York: Academic Press.

Kennedy, Christopher & Louise McNally. 2005. Scale structure, degree modification, and the semantics of gradable predicates. *Language* 81(2). 345–381.

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. 1981. The notional category of modality. In *Words, worlds, and contexts: New approaches in word semantics, ed. by hj eikmeyer and h. rieser*, 38–74. Berlin: Walter de Gruyter.

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. 2000. The event argument and the semantics of verbs, chapter 2. Manuscript. Amherst: University of Massachusetts, Amherst, MA. `http://semanticsarchive.net/Archive/GU1NWM4Z/`.

Krifka, Manfred. 1986. *Nominalreferenz und Zeitkonstitution. Zur Semantik von Massentermen, Pluraltermen und Aspektklassen.* Munich, Germany (published 1989): Fink.

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.

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. 1996. Plurality. In Shalom Lappin (ed.), *Handbook of contemporary semantic theory*, 425–457. Oxford, UK: Blackwell Publishing.

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.

Link, Godehard. 1983a. 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.

Link, Godehard. 1983b. The logical analysis of plurals and mass terms: A lattice-theoretical approach. In Reiner Bäuerle, Christoph Schwarze & Arnim von Stechow (eds.), *Meaning, use and interpretation of language*, 303–323. Berlin, Germany: de Gruyter.

Matthewson, Lisa. 2006. Temporal semantics in a superficially tenseless language. *Linguistics and Philosophy* 29. 673–713.

McCoard, Robert W. 1979. *The English perfect: Tense-choice and pragmatic inferences.* Amsterdam: North-Holland Press.

Montague, Richard. 1973. 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. 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*, 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.

Muskens, Reinhard. 1995a. *Meaning and partiality*. Stanford, CA: CSLI Publications.

Muskens, Reinhard. 1995b. Tense and the logic of change. In Urs Egli, Peter E. Pause, Christoph Schwarze, Arnim Von Stechow & Gotz Wienold (eds.), *Lexical knowledge in the organization of language*, 147–183. Amsterdam: John Benjamins.

Muskens, Reinhard. 1996. Combining Montague semantics and discourse representation. *Linguistics and Philosophy* 19. 143–186.

Muskens, Reinhard. 2005. Sense and the computation of reference. *Linguistics and Philosophy* 28(4). 473–504.

Muskens, Reinhard. 2011. A squib on anaphora and coindexing. *Linguistics and Philosophy* 34(1). 85–89. doi:10.1007/s10988-011-9091-8.

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. 1990a. *Events in the semantics of English*. Cambridge, MA: MIT Press.

Parsons, Terence. 1990b. *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. 1984. Compositionality. In Fred Landman & Frank Veltman (eds.), *Varieties of formal semantics*, 281–312. Dordrecht: Foris.

Partee, Barbara. 2006. Lecture 1: Introduction to formal semantics and compositionality. Lecture notes for Ling 310 The Structure of Meaning.

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.

Penka, Doris. 2016. Negation and polarity. In Nick Riemer (ed.), *The routledge handbook of semantics*, 303–319. Routledge.

Poesio, Massimo & Alessandro Zucchi. 1992. On telescoping. In *Proceedings of salt ii*, 347–366.

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.

Reichenbach, Hans. 1947. *Elements of symbolic logic.* Macmillan.

Romero, Maribel. 2008. The temperature paradox and temporal interpretation. *Linguistic Inquiry* 39(4). 655–667.

Russell, Bertrand. 1905. On denoting. *Mind* 14. 479–93.

Russell, Bertrand. 1910. Knowledge by acquaintance and knowledge by description. *Proceedings of the Aristotelian Society* 11. 108–128.

Sapir, Edward. 1944. Grading: A study in semantics. *Philosophy of Science* 11. 93–116.

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, 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.

Siegel, Muffy E. 1976. *Capturing the adjective*: University of Massachusetts, Amherst dissertation.

Smith, Carlota S. 1997. *The parameter of aspect.* Dordrecht: Kluwer.

Solan, Lawrence M. & Peter M. Tiersma. 2014. *Speaking of crime: The language of criminal justice.* University of Chicago Press.

Stalnaker, Robert. 1978. Assertion. In *Syntax and semantics*, vol. 9, Academic Press.

Strawson, P. F. 1950. On referring. *Mind* 59(235). 320–344.

Strawson, Peter. 1964. Identifying reference and truth-values. *Theoria* 30(2). 96–118.

Vendler, Zeno. 1957. Verbs and times. *Philosophical Review* 66. 143–160.

von Fintel, Kai & Irene Heim. 2011. Intensional semantics. MIT lecture notes.

von Stechow, Arnim & Atle Gronn. 2013a. Tense in adjuncts part 1: Relative clauses. *Language and Linguistics Compass* 7. 295–310.

von Stechow, Arnim & Atle Gronn. 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. doi: Alsoknownas{\emTractatusLogico-Philophicus}.

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.

Zwarts, Frans. 1995. Nonveridical contexts. *Linguistic Analysis* 25. 286–312.