Invitation to Formal Semantics

(Formerly known as *Semantics Boot Camp*)

Elizabeth Coppock
&
Lucas Champollion

Draft of August 21, 2020

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.

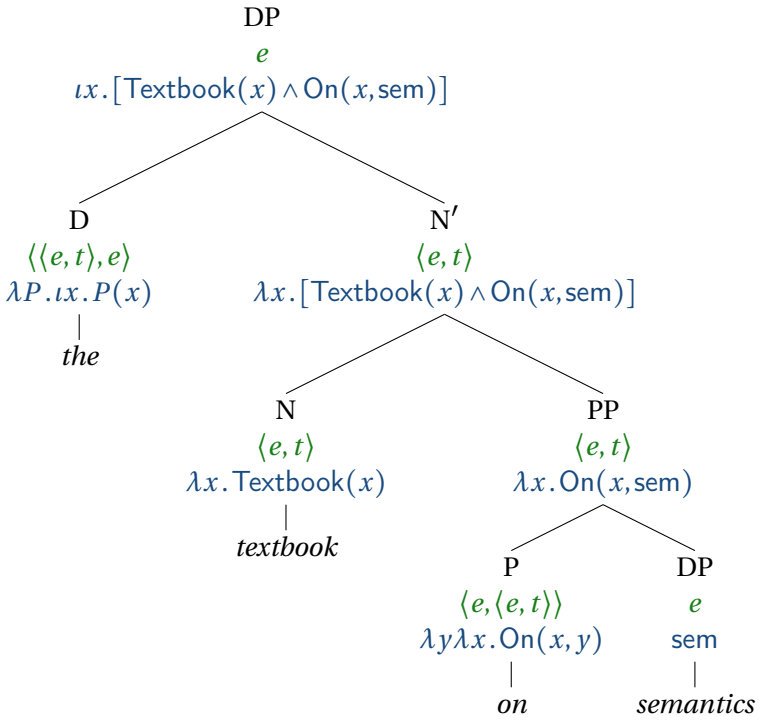
Students are guided through the development of a formally precise, compositional, model-theoretic account of semantics, using a logical representation language that is well-rooted in intellectual tradition, yet modern. The book familiarizes students with the main tools and techniques they need to understand 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, and Alex Warstadt, 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 creating the answer keys.

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

Contents

1 Introduction

1.1 Entailment

This is a book about meaning. What is meaning? Clearly, meaning is 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, in some sense. Case in point: one can type in "350 USD in SEK" and get back "3062.33 Swedish Krona" as the first result. This appears to be at least a step toward understanding.

On the other hand, computers are not very good at drawing INFERENCES from texts, and this suggests they don't understand language very well after all. 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 Portman does not speak Russian—if she did, then Russian would have been listed among the languages she speaks. The sentence would not be *false*, strictly speaking, if it turned out that Natalie Portman also speaks Russian. Still, it *somehow* implies that she does not. But when we ask Google whether Natalie Portman speaks Russian,

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

we do not get 'no' as the answer.

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

(2) Natalie Portman speaks more than two languages.

Sentence (2) necessarily follows from sentence (1). If (2) were false—in other words, if Natalie Portman spoke two languages or fewer—then (1) would certainly be false as well. But if we ask Google whether Natalie Portman speaks more than two languages, we do not get 'yes' as the answer.

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.

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

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

These are not the same kind of implication, but they can both be classified under that broader umbrella.[2] 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 listener will conclude that B is probably true (assuming that they trust the

[2]Terminological note: An IMPLICATION (or IMPLICATION RELATION) is a relation that holds between a premise and an implied conclusion. The noun *inference* normally describes the act of inferring a conclusion from a premise, but *inference* can also be used to mean *implication*. The verb *infer* is totally different from the verb *imply*, though; an intelligent person *infers* a conclusion from a premise, but a premise *implies* a conclusion. 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*.

speaker).'[3] This notion covers a wide range of subtypes of implication, including ENTAILMENTS, IMPLICATURES, and PRESUPPOSITIONS. This chapter provides a brief introduction to all three, and gives an overview of how, and to what extent, our theory of semantics will handle them.

1.2 Varieties of consequence

1.2.1 Defining entailment

Entailment is closely connected with reasoning. Somebody who infers (2) from (1) reasons correctly. Somebody who infers from

(3) Some lizards are pets.

that

(4) Some pets are lizards.

reasons correctly as well. But somebody who reasons from

(5) All cats are animals.

to

(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). But sentence (5) does not entail sentence (6).

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

(7) a. All men are mortal.

[3] This way of defining "implies" is used for eliciting human judgments in order to establish a gold standard in the 'Recognizing Textual Entailment' task, a branch of natural language processing that aims at developing computer systems that can capture inferences in natural language.

b. Socrates is a man.
c. ∴ Socrates is mortal.

The symbol ∴ is pronounced "therefore".

The reasoning in (7) is an ARGUMENT, in the sense that it presents a CONCLUSION as a consequence derived on the basis of one or more PREMISES.[4] In this case, the two premises are (7a) and (7b), and the conclusion is (7c). Arguments whose premises entail the conclusion are called VALID; others INVALID. In this book, we use the symbol ∴ for valid arguments and the symbol ∴̸ for invalid arguments.

What, exactly, is it to reason correctly or incorrectly? In other words, what makes an argument valid or invalid? Consider again this invalid argument:

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

The premise of this argument is true, but its conclusion is false. In general, whenever that is the case, an argument is invalid. But there are also invalid arguments with true premises and true conclusions:

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

Both premises of this argument are true, and so is its conclusion. But this is still not correct reasoning. The conclusion doesn't *follow* from the premises. We would not want to call this a valid argument.

[4] 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*.

How, then, can we tell if an argument is valid? What does it mean for a conclusion to follow from premises? What is it to reason correctly? One strategy for answering this question is based on considering what would have been the case if things had been different. Suppose that all cats were white, but that there were still black animals around, say black dogs. Then it would still be true that all cats are animals, and that some animals are black, so Premise 1 and 2 of (9) would be true. But now the conclusion would be false.

In general: To reason correctly, it is not enough if the premises and conclusion to all happen to be true in reality. It also needs to be the case that the conclusion would still be true *even if things had been different*, as long as the premises would still be true. This brings us to our definition of a valid argument. An argument is VALID if and only if: *Under all circumstances under which the premises would be true, the conclusion would be true too.*[5]

An argument can be valid, then, even if it has false premises. Here is an example:

(10) a. Lemonade is made from watermelon.

[5]Another way to characterize a valid argument is as one that has a correct form and does not depend on the meaning of content words like *cat*, *animal*, and *black*. The argument in (9) does depend on the meanings of these words. To see this, consider first what happens when these words are replaced by others:

(i) a. All triangles are shapes. (Premise 1)
b. Some shapes are round. (Premise 2)
c. ∴ Some triangles are round. (Conclusion)

This new argument has the same form as that in (9) and its premises are still true, but this time the conclusion is false. Now suppose we read the original argument in (9) while taking the words *cat*, *animal*, and *black* to mean what *triangle*, *shape*, and *round* mean. Read in that way it too has true premises but a false conclusion. The argument in (7), by contrast, will always have have true premises and a true conclusion no matter how we imagine the meanings of its content words to change. In this sense, it does not depend on their meaning. Validity in this sense is called 'formal consequence', as opposed to 'material consequence' as we have characterized it in the main text.

b. Watermelon is a type of vegetable.
c. ∴ Lemonade is made from vegetables.

Of course lemonade is not actually made from watermelon, and watermelon is not actually a vegetable. But still the conclusion follows from the premises. Here is a way to see it. If the premises *were* true, then the conclusion would be true as well no matter what else was different. And that is all it takes for the argument to be valid. An argument that is valid and whose premises are furthermore true is called SOUND. So the argument in (10) is valid but not sound.

Here is another example of an argument that is valid but not sound. This argument has a true and a false premise and a true conclusion:

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

If lemonade was made from watermelon, then so long as watermelon was still a type of fruit, lemonade would still be made from fruit.

We have said that a valid argument is one whose conclusion is entailed by the set of its premises, and we have characterized a valid argument as one in which the conclusion would be true in any circumstance where all the premises are true. Entailment can thus be defined as follows:

(12) A sentence A ENTAILS a sentence B if and only if:
There are no circumstances under which A is true but B is not true.

The intuition here is that whenever A does not entail B, one can point to a counterexample, that is to say, a circumstance in which A is true but B is not true. Entailment is the absence of such counterexamples.

Here is another way to think about what (12) says: It is *impossible* for there to be a circumstance in which A is true and B is not true. Therefore, if A entails B, then to say A and then deny B is to contradict oneself. Consider:

(13) Shawn ate a strawberry but Shawn did not eat a berry.

Anyone who made this series of assertions would be guilty of self-contradiction, because *Shawn ate a strawberry* entails *Shawn ate a berry*. In fact, in the next section, we will rely on this property of entailment in order to distinguish it from implicature.

Yet another way to define the conditions under which A entails B—a way that is simpler, though somewhat less precise—is as follows: *Whenever A is true, B is true, too.* Or, even more concisely: *The truth of A guarantees the truth of B*. All of these get at the same idea.

Notice that in (12), entailment is defined in terms of truth, and truth is relative to a circumstance (we say 'a circumstance in which A is true'). A given sentence might be true in circumstance, but false in another. The set of circumstances in which a given sentence is true can also be called the TRUTH CONDITIONS of a sentence. Given an association between sentences and their truth conditions, it is possible to determine which pairs of sentences stand in an entailment relation. The theory we develop in this book will do just that.

Exercise 1. The definition in (12) applies to the case where we have one premise and one conclusion. Write a corresponding version of this definition that applies to the case when we have multiple premises and one conclusion, and give a definition of a valid argument in terms of entailment.

Exercise 2. 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) $2 + 2 = 4$; therefore, Paris is the capital of France.

(c) There is no mammal that does not have lungs; Angela Merkel is a mammal; therefore, Angela Merkel has lungs.

(d) Copenhagen is either in Denmark or in the Netherlands; it's not in the Netherlands; so it is in Denmark.

Exercise 3. 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.

Importantly, in order to determine whether a given argument 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:

(14) 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:

(15) a. Florida is north of no U.S. state.
b. No U.S. state is north of Alaska.
c. ∴ 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 (16a) were true. Does it follow that (16) is true?

(16) 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. (16) can be read either as saying that there is no email Jane has responded to, or that there is one particular email that she hasn't responded to. This kind of ambiguity is known as a SCOPE AMBIGUITY, and the formal tools that we will develop in this book will help to elucidate the various readings that sentences like this can have. LEXICAL AMBIGUITY is when a word has multiple senses, and this can also muddy the question of whether one sentence entails another. In fact, we will see this in the next example.

Consider the following two arguments, which are identical in syntactic structure:

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

versus:

(18) a. Sue and Martha are vegetarians.
b. #. Sue is Martha's vegetarian.

There is at least one reading of (17) on which it is valid, but there is no reading of (18) on which it is valid. (The validity of (17) depends on whether 'Sue and Martha are sisters' means '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.

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. The 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 Entailment vs. implicature

We now discuss how to distinguish entailments from another kind of implication, namely implicatures. Recall example (1), repeated here as (19):

(19) 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 the Russian diplomat Sergey Kislyak in perfectly fluent Russian. Would you conclude, based on this information, that (19) 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 (19) knew this, they would be "lying by omission". 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 are the Maxim of Quantity, which requires that speakers provide as much information as needed for the information exchange. 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. Along with the Maxim of Quantity, Grice posits a Maxim of Relevance, which enjoins speakers to be

relevant, which in more modern terms is to address the question under discussion. As illustrated by this example, both the Maxim of Quantity and the Maxim of Relation play a role in how implicatures arise.

Exercise 4. What is the difference between an *implication* and an *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. Unlike entailments, implicatures can be CANCELLED without producing a contradiction. For example, one could say, without contradicting oneself:

(20) 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:

(21) 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. 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*. In general, if you know that sentence *A* implies sentence *B* but you don't know whether this implication is an entailment or an implicature, you can run this DEFEASIBILITY TEST by constructing an example in which *A* is followed by the negation of *B*, and observing whether the result is contradictory. If so, then the implication is not defeasible, and thus you have an entailment on your hands.

More specifically, to run the defeasibility test for an implication from sentence *A* to sentence *B*, the first step is to construct a text of the form *A* & *not-B*, where *not-B* negates *B*, and & is the most appropriate conjunction (*but*, *and*, *in fact*, whichever fits best).[6]

Constructing a negation can sometimes be a bit tricky. It's not always a matter of just adding a *not*. For example, let's consider whether (22a) entails (22b).

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

To run the defeasibility test on this example, we have to construct a negation of (22b). In this case, rather than adding a *not*, as we

[6]Another way of saying '*not-B* negates *B*' is '*not-B* is a negation of *B*'. We say *a negation* rather than *the negation* because in principle there may be many sentences that count as negations of a given sentence, although they will all be equivalent to each other.

As we will discuss later, sentences express *propositions*. For a given *proposition*, there is only one *proposition* that can be seen as its negation: the proposition that is true whenever the original proposition is false. But since there are many ways of expressing the same proposition, there's no one unique sentence that negates another given one.

just did, we can take one away: *All Republicans voted 'yes'*. Now we can ask whether *A & not-B* is SELF-CONTRADICTORY:

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

Would someone who uttered (23) be contradicting themselves? No. So *A & not-B* is not self-contradictory in this case. So the implication from (22a) to (22b) is not an entailment; it's a conversational implicature.

Now, consider the following sentence:

(24) Everybody likes chocolate.

Which of the following negates (24)?

(25) a. Nobody likes chocolate.
b. Not everybody likes chocolate.

Answer: Of these two, the sentence that negates (24) is (25b): *Not everybody likes chocolate*. The other one, *Nobody likes chocolate*, says something stronger than than the one that negates it. In principle, there are three types of circumstances:

(i) Everybody likes chocolate.

(ii) Some people like chocolate and some people don't.

(iii) Nobody likes chocolate.

The sentence *Not everybody likes chocolate* is true in (ii) and (iii), while the sentence *Nobody likes chocolate* is only true in (iii). So *Nobody likes chocolate* is true in fewer types of circumstances, and in that sense it makes a stronger statement. In (ii), neither (24) nor (25a) is true; it's not the case that everyone likes chocolate, and it's also not the case the nobody likes chocolate. Neither is true. This is characteristic of CONTRARY OPPOSITION, as opposed to CONTRADICTORY OPPOSITION. These two terms are defined as follows:

(26) A sentence *A* is stands in CONTRADICTORY OPPOSITION a sentence *B* if and only if:
A and *B* cannot be true in the same circumstance, and they cannot be false in the same circumstance.

(27) A sentence *A* is stands in CONTRARY OPPOSITION a sentence *B* if and only if:
A and *B* cannot be true in the same circumstance but they can be false in the same circumstance.

Let us consider what relation holds between (24) *Everybody likes chocolate* and (25a) *Nobody likes chocolate*: These two sentences cannot be true at the same time, but they *can* be false at the same time, as we have just seen. These two sentences therefore stand in *contrary opposition*. The sentence that stands in *contradictory opposition* to (24) *Everybody likes chocolate* is (25b) *Not everybody likes chocolate*. If the one is true, then the other must be false, and vice versa.[7] We define a NEGATION of a sentence as one that stands in *contradictory* opposition to it.

As another example, consider the following sentence:

(28) Jo is tall.

A sentence that stands in contradictory opposition to (28) 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 (28). 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

[7] The distinction between contrary and contradictory opposition goes back to Aristotle and plays an important role in the square of opposition of traditional medieval logic. This fascinating topic is described, for example, in Parsons (2017).

e.g. Kennedy & McNally 2005).

Again, a negation of a sentence stands in contradictory, rather than contrary opposition to it, so to check whether you have constructed your negation correctly, you can check what type of opposition your sentences stand in to each other. But if you are faced with a really tricky case, you can always just add *It is not the case that* to the beginning of the sentence; *It is not the case that S* is essentially guaranteed to be the negation 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*; when one is true, the other is true too, and when one is false, the other is false too. Thus, even if you don't use the *It is not the case that* strategy, another way to check whether you have constructed the negation properly is to ask yourself whether your sentence is equivalent to the *It is not the case that* version.

Once you have constructed the negation, it is important that you pose the right question about the example of the form *A & not-B* to a native speaker of the language (possibly yourself, if you are a native speaker of the language). The question is not 'Is this example grammatical?' or 'Is this example true?' but rather '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 5. 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.

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.

Another test that can be used to distinguish between entailments and implicatures is called the REINFORCEMENT TEST. The

idea behind it is that if A entails B, then saying B after one has just said A sounds redundant, because B was just directly implied. Consider the following contrast:

(29) She speaks English, Hebrew, Spanish, German, Japanese, and French; { #and / #but / #in fact } she speaks more than two languages.

(30) She speaks English, Hebrew, Spanish, German, Japanese, and French; but she doesn't speak Russian.

In both cases, we have a sentence of the form 'A & B', where B is implied by A. But in the first case, B is entailed by A, so 'A & B' sounds redundant. In the second case, B is merely conversationally implicated by A, so 'A & B' does not sound redundant. The kind of observation to be made about the constructed example is different here: Rather than asking whether the example sounds contradictory, we ask whether it sounds redundant.

To summarize, we have presented two tests for distinguishing between entailments and implicatures. For the DEFEASIBILITY TEST, the example to construct is of the form 'A & not-B', and the question to ask is whether it sounds contradictory. If yes, then the test suggests that the implication from A to B is an entailment. For the REINFORCEMENT TEST, the example is of the form 'A & B', and the question to ask is whether it sounds redundant. If yes, then the test suggests that the implication from A to B is an entailment.

Exercise 7. Consider the following pairs of sentences:

(31) a. Every dog barked.
b. Every labrador barked.

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

(33) a. Sam regrets winking at Dave.

b. Sam winked at Dave.

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

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

(36) 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 and reinforcement tests.

1.2.3 Entailment vs. presupposition

Normally, when a sentence is not true, its negation *is* true. For example, the following sentence is *not* true:

(37) In Stockholm, January is the warmest month of the year.

while its negation *is* true:

(38) In Stockholm, January is *not* the warmest month of the year.

But it can happen that neither a sentence nor its negation is true. Consider the following sentence:

(39) The theremin duo that Mozart wrote is very famous.

As it happens, Mozart died long before the theremin was invented, and therefore could never have written any piece for theremin, let alone a duo. So this sentence is certainly not true. And yet its negation is not true either:

(40) The theremin duo that Mozart wrote is not very famous.

The culprit behind this odd state of affairs is the DEFINITE DESCRIPTION *the theremin duo that Mozart wrote*. To a first approximation, a DEFINITE DESCRIPTION is a phrase of the form *D* + *X*, where *D* is the DEFINITE ARTICLE (*the* in English), picking out the unique individual that satisfies the description *X*. By use of the definite description in (40), a speaker becomes committed to the existence of theremin duos written by Mozart, whether the sentence that the definite description is embedded in contains negation. Both (39) and (40) entail (41).

(41) Mozart wrote a theremin duo.

Since both sentences entail something false, neither sentence is true.

The type of implication involved here is not an ordinary entailment, but it is not a conversational implicature either; it is a PRESUPPOSITION. Presupposition can be defined either as something that speakers do or as something that sentences do. In the former sense, to presuppose something is to take it for granted, to treat it as uncontroversial and known to everyone participating in the conversation. When a *sentence* presupposes something, it signals that such a presupposition is made by the speaker in contexts where the sentence is appropriately used.

What we have just given are PRAGMATIC DEFINITIONS 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 (41) is not true, (39) 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 definite article *the* triggers a presupposition of existence. Another example of a presupposition trigger is the adverb *still*. For example, if I said (42), I would signal (43) through a presupposition.

(42) Natalie Portman still speaks French.

(43) 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 *Natalie Portman no longer speaks French* is true, *Natalie Portman spoke French at some time in the past* 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. If I were to negate (42) as follows (slightly awkward, but entirely comprehensible):

(44) Natalie Portman doesn't still speak French.

then I would still be implying that Natalie Portman 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.

(45) 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:

(46) 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,

(47) Does Natalie Portman speak French?

they would not be implying that Natalie Portman spoke French, of course. And yet:

(48) 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 (49a) implies (49b), broadly speaking; anyone who heard (49a) would certainly conclude that (49b) is true, assuming they trusted the speaker.

(49) 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 (49a) in an entailment-cancelling environment, such as negation, the antecedent of a conditional, or a yes/no question. Let's try all three, just to be on the safe side:

(50) 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. **Yes/no question**
Does Kim's twin sister live in Austin?

These sentences all imply that Kim has a twin sister. So the inference projects.

The decision procedure for distinguishing between the various types of implication relations is summarized in Figure 1.1. Use the defeasibility test to distinguish between entailment and implicature; use the projection test to distinguish between ordinary entailment and presupposition.

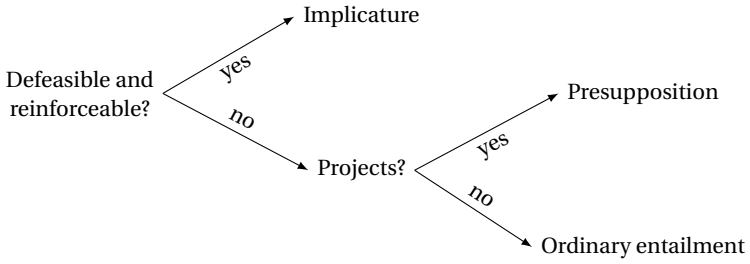


Figure 1.1: A decision tree for categorizing implications

Exercise 8. 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 9. Consider the following two sentences.

(51) 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 (52a), but the *succeed* sentence implies that he did (52b), and the *fail* sentence implies that he did not (52c).

(52) 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:

(53) a. (51a) 'succeed' → (52a) 'try';
b. (51b) 'fail' → (52a) 'try';
c. (51a) 'succeed' → (52b) 'did';
d. (51b) 'fail' → (52c) '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 pre-

supposition) 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*.

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 associated with denotations. 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 circumstance where A is true, but B is not. In order to explain entailments, therefore, we will define an association between sentences and truth conditions, that is, sets of circumstances. 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 Wittenstein, Donald Davidson, David Lewis, and Richard Montague.

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 circumstances, 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 they determine what would have to be in 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.

As a sentence may be true in one circumstance but false in another, the truth of a sentence depends on the circumstance relative to which it is evaluated. What exactly is a circumstance? For the purposes of this book, we see circumstances as POSSIBLE WORLDS. A POSSIBLE WORLD can be thought of as "a complete specification of how things are, or might be, down to the finest semantically relevant detail" (Dowty et al., 1981, 12). (The world in which we live is the ACTUAL WORLD. For instance, the sentence *Beyoncé is an astronaut* is false in the actual world.)[8]

Exercise 10. 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

[8] An alternative would be to construe a circumstance as a STATE OF AFFAIRS, which "is a less comprehensive or detailed notion than a 'possible world,' since a state of affairs may concern only some aspects of a world." (Dowty et al., 1981, 12). The term SITUATION can be used in this sense as well, among others (Kratzer, 2019). In this book, we will treat circumstances as possible worlds, as opposed to, say, situations, for a couple of reasons. First, it is uncontroversial how to define logical notions such as negation in the possible worlds setting. Second, this is the approach that is generally taken in the formal semantics literature when the issue under consideration is not how to treat circumstances *per se*, and in that sense it is the more 'vanilla' choice.

true."[9] 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.

Certainly, there is more to meaning than truth conditions. But when we talk about meaning, we can distinguish between the meaning of a SENTENCE and the meaning of an UTTERANCE. A sentence is a particular type of word sequence that could in principle be used on many different occasions, or on none; an utterance on the other hand is a particular event of speech, located in time and space. An utterance is 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. The implicatures that an utterance gives rise to in context can be seen as part of its meaning, so truth conditions are not all there is to meaning, at least for utterances.

But even *sentence meaning* goes beyond truth conditions. For one reason, not all sentences have truth conditions, arguably. 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 sentences that arguably do not 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 developed for this purpose can profitably be extended to a wider range of sentence types once they are in place.

Even the meaning of declarative statements of fact goes beyond truth conditions. For one example, consider *John is married* vs. *John has a spouse*. These two sentences have the same truth

[9] 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.

conditions, but differ in how easy they make it to refer to John's spouse with a pronoun in the next sentence. Consider the following contrast, where the question mark indicates that the example is of 'questionable' acceptability.[10]

(54) a. John is married. ?She is coming.
b. John has a spouse. She is coming.

Another famous example of this kind is due to Barbara Partee (cited in Heim 1982b):

(55) 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. We will explain this point in the next section, after we have introduced the more fundamental distinction between INTENSION and EXTENSION.

Exercise 11. In what ways does meaning go beyond truth conditions? Discuss three.

[10] Other so-called DIACRITICS of this kind include the asterisk (*), which indicates that the sentence is ungrammatical, and the hash-mark (#), which indicates that the sentence is semantically anomalous—makes no sense—or pragmatically infelicitous (inappropriate) in a given context.

1.3.2 Compositionality

The systems for natural language understanding that we develop here 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 that defines a concept or a procedure in terms of itself, without being circular. We will illustrate how the concept of recursion applies in our case when we present the systems in detail.) In this respect, the systems you will see here will 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 illustrate what it means for a semantic theory to be compositional. For the purposes of discussion 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. 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 12. What would it look like if the principle of substitutivity was violated? Give an example from math.

Exercise 13. What does it mean for a theory of meaning to be *compositional*?

Let us be a bit more specific about the sense in which the "meaning" of a larger expression is determined by the meanings of the parts. We will define a set of rules that assign a SEMANTIC VALUE, or equivalently, DENOTATION, to each grammatical expression of the language, relative to a given circumstance of evaluation. Following Frege (1892 [reprinted 1948]), we assume that the denotation of a declarative sentence is a TRUTH VALUE: True or False (sometimes represented by the numbers 1 and 0). Some systems allow for three, four, or even infinite truth values, but we'll start with just two, True and False. If a given sentence is true in a given circumstance, then we will say that its denotation relative to that circumstance is the truth value True. Otherwise, its denotation is False (unless the sentence contains a false presupposition, and we have a third truth value Neither at our disposal; then the denotation may be this third truth value). The truth conditions of a sentence can be viewed as a mapping that associates each possible circumstance with the truth value that the sentence has in that circumstance. An association between circumstances and truth values of this kind is sometimes referred to as a PROPOSITION, because in some sense it encapsulates the meaning of a claim that can be true or false.

Sentences *denote* truth values, and *express* propositions. So what is the meaning of a sentence? Is it a truth value, or proposition, or something else? It depends what we mean by "meaning". Rudolf Carnap distinguished between two levels of meaning, called INTENSION and EXTENSION. (Note that the *s* is not a typo in

these words; there is a big difference between *intension*, with an *s*, and *intention*, with a *t*.) The extension of a declarative sentence at a given circumstance is a truth value; its intension is the proposition it expresses.

Intensions and extensions are not just for sentences; words and phrases also have both intensions and extensions. A general way of defining INTENSION is as a mapping that associates any given circumstance to its extension at that circumstance. With this definition in hand, we can talk about the intensions and extensions of all different kinds of expressions, such as noun phrases like *the king of Sweden*. Relative to the actual world, the extension of *the king of Sweden* is the individual who is in fact the king of Sweden. The intension is a function that maps any given circumstance to the individual that is the king of Sweden in that circumstance. (We'll discuss what to do when there is no such individual, as in the case of *the theremin duo that Mozart wrote*, in Chapter 8.) Nouns like *dog* and *theremin* also have intensions and extensions. The extension of *dog* in the actual world is the set of dogs that actually exist; the intension is a mapping that associates each possible circumstance with the set of dogs in that circumstance. For a sentence, the intension is a mapping that associates circumstances with truth values. (This way of characterizing intensions becomes indispensable when the set of truth values grows beyond just True and False, as in three-valued logic.)

Notice that two expressions with the same extension at a particular world could have different intensions. Here is an example from Quine (1951): It just so happens that every species with a heart also has a kidney, in the actual world. So *creature with a heart* has the same extension as *creature with a kidney* — these two expressions pick out the same set of species in the actual world. But in principle, there could be a world in which there are species that have hearts, but no kidney, or vice versa. So these two expressions do not have the same intension. For another example, any two sentences that both happen to be true have the same ex-

tension (the truth value True), but they might differ in their intension. Take for example *Paris is the capital of France* and *Berlin is the capital of Germany*. These are both true in the actual world (at the time of writing), but things could have been otherwise, so that one of these sentences is true and the other is false. Hence the two sentences have different intensions. In general, intensions make finer-grained distinctions than extensions.

HYPERINTENSIONALITY is a phenomenon in which two expressions have the same intensions, and yet they *still* differ in meaning. For example, any two sentences that express mathematical truths will be true in *all* possible worlds. The sentence $2 + 2 = 4$ is true under just the same possible worlds as *There is no largest prime number* is true (namely all of them, since they are mathematical truths). So their truth conditions are the same. And yet their meaning is different, as shown by the fact that the following two sentences can differ in their truth value:

(56) a. Sandy knows that $2 + 2 = 4$.
b. Sandy knows that there is no largest prime number.

There could be a situation where (56a) is true but (56b) is false. It seems reasonable to assume that if two sentences lead to two different sets of truth conditions when embedded in the complement of *know*, then they have different meanings. Given this assumption, the fact that (56a) and (56b) have different truth conditions shows that $2 + 2 = 4$ and *There is no largest prime number* differ in meaning, although their intensions are the same. We will not have much to say about hyperintensionality in this book, although it is an important and ongoing topic of research (see Jespersen & Duží 2015 for an overview).[11]

We said at the beginning of this subsection that the systems for

[11] Hyperintensionality has been linked to Frege's (1892 [reprinted 1948]) distinction between SENSE and REFERENCE. Frege observed that although *the morning star* and *the evening star* both refer to the same object (namely the planet Venus, which incidentally is not a star at all), (ia) differs in meaning from (ib).

natural language understanding that we develop here 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). Armed with the concepts of intension and extension, we can now be a bit more specific about what we mean when we say 'meaning'. The first formal systems that we develop in this book will be built on the assumption that the extension of a complex expression at a particular circumstance depends solely on the extensions of the parts at that circumstance. But in some cases, when two expressions combine, the extension of the whole in a particular circumstance depends on the *intension* of at least one of the parts. A purely EXTENSIONAL SYSTEM is one in which the extension of an expression depends solely on the extensions of the parts, while an INTENSIONAL SYSTEM is one in which the extension of an expression may depend on the intension of one of the parts (cf. Gamut 1991, Vol. 2, p. 5, who writes that a system is extensional if expressions with the same extension may be "freely substituted" for each other). In the chapter on intensional semantics, we will define a system in which the extension of a larger expression can depend on the intension of the parts.

Now, *how* is the meaning of a complex expression determined from the meanings of the parts? Frege (1891) proposed that semantic composition of two parts involves 'saturation'; one part is

(i) a. The morning star is the evening star.
b. The morning star is the morning star.

In principle it would be possible to learn something from (ia), but (ib) is entirely uninformative. So although there is some level at which *the morning star* has the same meaning as *the evening star*, there must be another level at which these two differ in meaning. Frege called these two levels REFERENCE and SENSE, respectively; the two expressions have the same REFERENT (namely Venus), but they differ in their SENSE. Many of the things that we have said about hyperintensionality also apply to Frege's 'sense', while Carnap's 'extension' corresponds closely to Frege's 'reference'. Carnap's 'intension' falls somewhere in between Frege's 'sense' and Frege's 'reference'.

somehow missing something ('unsaturated'), and the other part is the missing piece (and thus 'saturates' 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.[12]

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 language in which we express 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 natural language (say, English), we must use a language in which to 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. In this book,

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

we are using English (with some mathematical notions mixed in) to theorize about English, so the meta-language is English (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.

Exercise 14. 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 French 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.

Our theory will thus consist of two mappings:

- a mapping from expressions of 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, 1973b).

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, 1970). That style is known as DIRECT INTERPRETATION, and it is adopted in the Heim & Kratzer (1998) textbook.

Let us consider an example. The English name *Natalie* will be translated into logic as natalie, which in turn has as its semantic value the individual Natalie Portman.

ENGLISH	LOGIC	SEMANTIC VALUE
Natalie	natalie	Natalie Portman

We refer to Natalie Portman in the meta-language using her name, "Natalie Portman". (We might as well have put a photograph of Natalie Portman instead, in order to drive home the point that it is really the individual herself that we mean to indicate.) In the system that we will set up, the English name *Natalie* indirectly becomes associated with this individual, by virtue of the fact that it is associated with the symbol natalie, which is part of the logic. This logic symbol natalie will in turn be associated with the individual Natalie Portman via the semantic rules we will develop in Chapter 4.

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 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 contradiction. 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 15. 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 it to suit your purposes. Accounting for a wider range of phenomena may well require fundamental changes to the theory, but if you understand this simple theory deeply enough to go in and adjust the foundations, then you will be in a position to make 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 *B* is true but *A* 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. 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 English; there are some dialects of English with so-called 'positive *anymore*', in which *anymore* is not restricted to negative or negative-like environments):

(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. But it's not just negative environments where NPIs can be found. Here is a sampling of the data (Ladusaw, 1980).

(2) { No one / At most three people / Few students / *Someone / *At least three people / *Many students } who had *ever* read *anything* about phrenology attended *any of the lectures*.

(3) I { never / rarely / seldom / *usually / *always / *sometimes } *ever* eat *anything* for breakfast *anymore*.

(4) a. Susan finished her homework { without / *with } *any* help.
b. Susan voted { against / *for } *ever* approving *any* of the proposals.

(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.

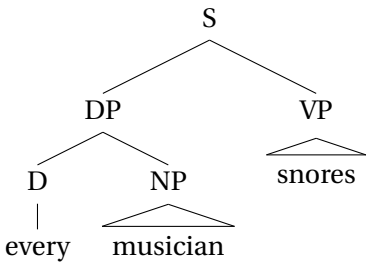
(6) It's $\left\{\begin{array}{l}\text{hard}\\ \text{difficult}\\ \text{*easy}\\ \text{*possible}\end{array}\right\}$ to find *anyone* who has *ever* read *anything much* about phrenology.

(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', a phrase headed by a noun) to form a DP (for 'determiner phrase', 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. (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].

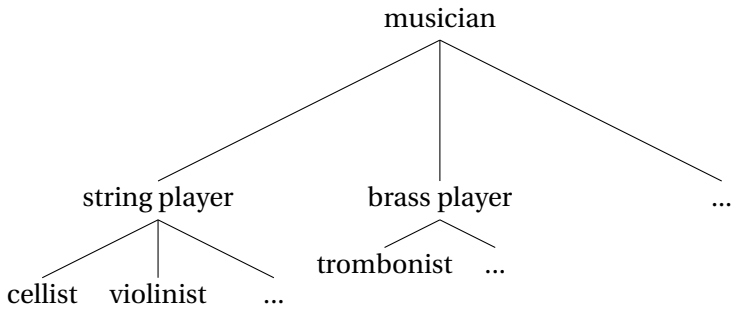
This shows that the ability to license negative polarity items is not a simple yes/no matter for each lexical item.

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 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.

[1] As discussed in Footnote (i) in Chapter 1, a distinction can be drawn between 'formal' and 'material consequence', although we are typically interested in material consequence in this book. The argument in (12) is 'materially valid' because it is impossible to 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 formally valid, because it is not valid solely in virtue of its form. Similar remarks apply to subsequent examples in this chapter.



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.

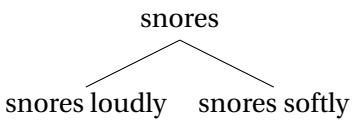
But negation and the determiner *no* reverse the entailment pattern; witness the validity of the following arguments:

(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.



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

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

But negation reverses the direction of entailment, so that the VP becomes 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 (relative to a given circumstance; the set of cellists in one circumstance may differ from the set of cellists in another). Regardless of circumstance, every cellist is a musician, so every member of the set denoted by *cellist* is a member of the set denoted by *musician*. Hence the denotation of *cellist* is always 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:

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

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:

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

[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 one:

$$\{3\}$$

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

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

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

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

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 ...

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

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

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

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".

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 $\{b\}$ is not the same thing as b. One is a set and the other might not be. (It depends whether b is a set; we have made no assumptions about what b is.) The following is a true statement, though:

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

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 (two sets are EQUAL iff they have the same members):

$A \subset B$ iff (i) for all x: if $x \in A$ then $x \in B$ 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:

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

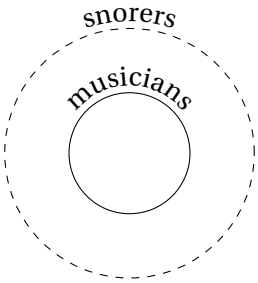
The reverse of subset is superset. A is a SUPERSET of B, written $A \supseteq B$, if and only if every member of B is a member of A.

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

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.

$A \supset B$ iff (i) for all x: if $x \in B$ then $x \in A$ 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:

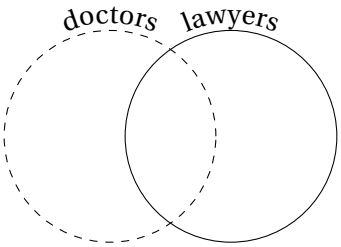
$$\{a, b, c\} \cap \{b, c, d\} = \{b, c\}$$

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

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

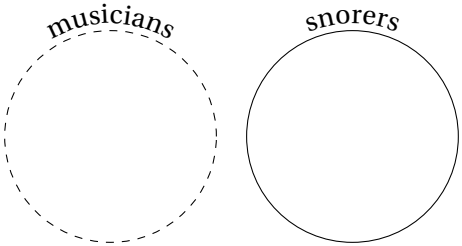
$$\{a, b\} \cap \{a, b\} = \{a, b\}$$

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 dashed circle in the following diagram represents the set of doctors, and the plain circle 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.)

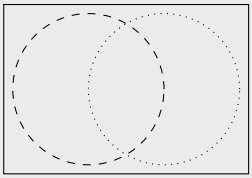


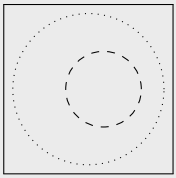
The English determiner *some* can be thought of in terms of intersection as well, as a relation between two sets X and Y which holds iff there is some member of X which is also a member of Y, i.e., iff the intersection between X and Y is non-empty. For instance, *some musician snores* is true iff there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

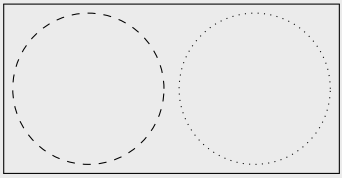
The determiner *no* can be thought of as a relation between two sets X and Y which holds if the two sets have no members in common, in other words, iff the *intersection is empty*. So *no musician snores* holds iff there is no individual who is both a musician and a snorer. In that case, the two sets 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 or in B (or both).

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

For example:

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

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

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

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:

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

(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:

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

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 $\{A, B\}$?

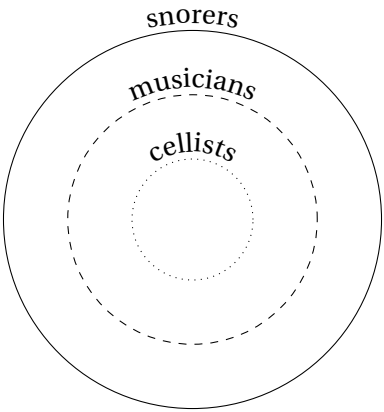
(i) Is a a member of $A \cup B$?

2.4 Negative polarity items revisited

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

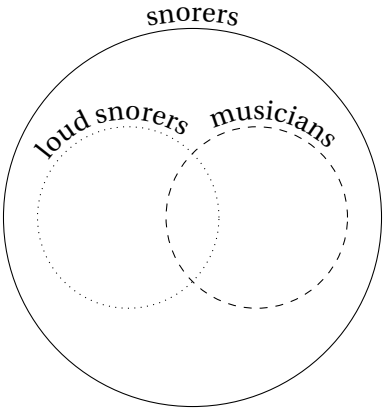
- *Every X Y* is true if and only if X is a subset of Y.
- *Some X Y* is true if and only if there is a non-empty intersection between X and Y.
- *No X Y* is true if and only if X and Y have an empty intersection.

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:



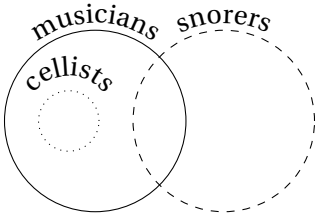
So, from *Every musician snores* it follows that *Every cellist snores*. In this particular example, we have taken X to be the set of musicians, X' the set of cellists, and Y the set of snorers. In general, *every* X Y entails *every* X' Y whenever X' is a subset of X. We say that *every* is LEFT DOWNWARD 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 δXY entails $\delta X'Y$ for all X' that are subsets of X.

A determiner δ is RIGHT DOWNWARD MONOTONE iff δXY entails $\delta XY'$ 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 52, 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) presup-

poses 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. 59), 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. The DOMAIN is the set of entities that occur as a first member, and the RANGE is the set of entities that occur as a second member. The set B is called the CODOMAIN. (We are not aware of any corresponding term for the set A.) Formally, a binary relation over A and B is a (proper or non-proper) subset of the Cartesian product $A \times B$.

The domain and range 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. In order for a relation to be reflexive, the domain must be a subset of the range – either equal to it or a proper subset of it.

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 gives 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.)

An example of a function is a relation that maps a person to their height in feet and inches. For example, given Michelle Obama it gives back $5'11''$ (five feet and 11 inches). Because functions are relations, a function is 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.

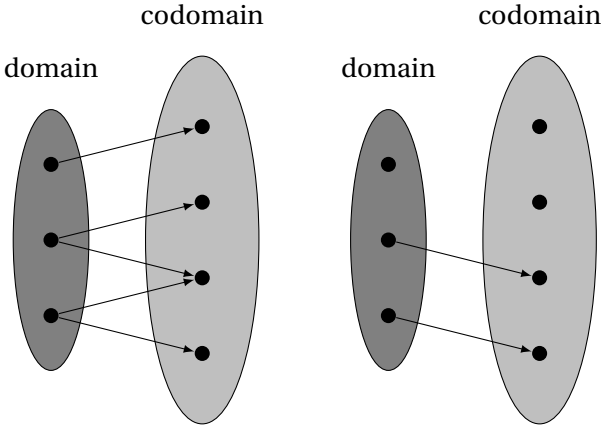


Figure 2.1: Two non-functions

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

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

or using large brackets like this:

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

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

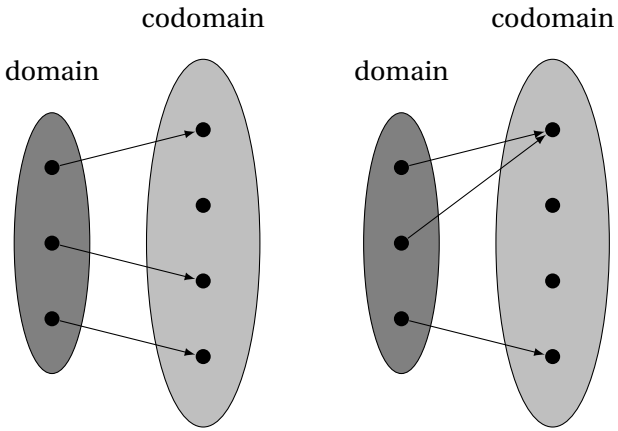


Figure 2.2: Two functions

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* 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 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 male(Björn)?

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

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

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 entailment is valid:

(1) All cars are blue.
∴ All German cars are blue.

It should also predict that the following are valid arguments:

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

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

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

We suggested a simple theory of entailment: sentences express propositions (construed as sets of circumstances), and entailment corresponds to the subset relation between propositions. In order to develop such a theory, we must specify a way to associate sen-

tences of natural language with the corresponding propositions.

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. [1] The first formal language we will consider is PROPOSITIONAL LOGIC. In propositional logic, we can express arguments like (2) but not like (3) or (4). We will then introduce PREDICATE 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 (2)) is clearly valid:

(5)	If it rained last night, then the lawn is wet.	(Premise 1)
	It rained last night.	(Premise 2)
	$\therefore$ The lawn is wet.	(Conclusion)

This argument has two premises and a conclusion. The conclusion follows logically from the collection of the premises: As long

[1] Typically, a logic consists of a language together with either a model-theoretic semantics, or a deductive system (i.e. a collection of inference rules, also called calculus), or both. Following most natural language formal semantics, this book focuses on the model-theoretic aspect of the logics it uses, and discusses inference rules only informally and in passing. Formal semantics that focuses on the deductive system is called PROOF-THEORETIC SEMANTICS; for a good introductory textbook, see for example Carpenter (1998).

as the premises are both true, the conclusion is true too. One might disagree with the premises, but as long as they are both granted, the conclusion follows. Hence, the argument is valid.

The argument in (5) is an instance of the argument form known as MODUS PONENS. Modus Ponens describes arguments matching the following general template:

(6) If P, then Q.
P.
$\therefore$ Q.

Exercise 1. Give another argument using Modus Ponens.

Now consider this superficially similar argument:

(7) 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 reason is that 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 necessarily true when both of the premises are true (that is, because the conclusion is not entailed by the premises taken together), the argument is not valid. This argument form is called AFFIRMING THE CONSEQUENT, and it is a well-known FALLACY—an argument form that is not valid. Affirming the consequent has the following general template.

(8) If P, then Q.
Q.
$\not\therefore$ P.

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

Propositional logic aims to capture the difference between valid argument forms and fallacies, where the templates involved have placeholders for entire sentences.

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 a simple proposition—roughly, the kind of thing that is expressed by a simple declarative sentence that does not contain any of the words *and, or, not, if, then*. For example, the propositional letter P could stand for the (false) proposition that Boston is the capital of Nebraska, or the (true) proposition that red is a primary color. 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. Typical choices besides ours are *p, q, r* and *a, b, c*. When more letters are needed, it is customary to use primes as in *P, P', P", 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 L_{Prop}.

Now, what is the denotation of a formula? Recall from Chapter 1 that Frege suggested that the extension of a natural language sentence is a truth value: True or False.[2] We also said that the truth value of a sentence depends on the circumstance relative to which it is evaluated. An INTERPRETATION FUNCTION, sometimes just called INTERPRETATION, for a given propositional language is a function that maps each propositional letter of that language to a truth value. Here is an example of an interpretation function for L_{Prop}.

$$\begin{bmatrix} P & \rightarrow & 1 \\ Q & \rightarrow & 1 \\ R & \rightarrow & 0 \end{bmatrix}$$

This says that P and Q are true, while R is false. An interpretation function characterizes a circumstance, by specifying which propositional letters are true and which are false. Formulas in propositional logic are assigned denotations relative to a given interpretation.

[2] Later, we will countenance a third truth value (Neither), but here we stick to just two.

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 is called a MODEL FOR that formula. Later, when we get to predicate logic, the notions of 'model' and 'interpretation' will be distinguished more sharply.

Formulas in propositional logic may be built up from smaller formulas, just as natural language expressions may be built up from smaller expressions. 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 True or False. 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, 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"). Meta-variables are not themselves part of L_{Prop};

they are part of the META-LANGUAGE that we use to talk about $\mathrm{L}_{\mathrm{Prop}}$ 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:

$$\begin{bmatrix} P & \to & 1 \\ Q & \to & 1 \\ R & \to & 0 \end{bmatrix}$$

Call this I_1. Then, for example, $[\![P]\!]^{I_1}$ ('the denotation of P under I_1') is 1. 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. I will typically differ from one propositional language to the other, while $[\![\cdot]\!]^I$ extends I in the same way for each propositional language.

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 $I(P) = 1$, then $[\![P]\!]^I = 1$ as well.

Exercise 4. 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:

(9) not [P or Q]. [not P] and [not Q]

as is its converse,

(10) [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':

(11) [[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:

(12) not [P and Q]
∴ [not P] or [not Q]

Its converse is valid as well:

(13) [not P] or [not Q]
∴ not [P and Q]

Again, we have an equivalence:

(14) [[not P] or [not Q]] if and only if [not [P and Q]]

The equivalences in (11) and (14) are called DE MORGAN'S LAWS.

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

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

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 should be 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. Again, we use the number 1 to represent True and the number 0 for False.

P	$\neg P$
1	0
0	1

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 $[\![\neg\phi]\!]^I = 1$ if $[\![\phi]\!]^I = 0$, and 0 otherwise.

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:

P = Susan volunteers on Monday
Q = Susan volunteers on Wednesday

We can 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	$P \wedge Q$
1	1	1
1	0	0
0	1	0
0	0	0

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 $[\![\phi \wedge \psi]\!]^I = 1$ if $[\![\phi]\!]^I = 1$ and $[\![\psi]\!]^I = 1$, and 0 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	$[P \vee Q]$
1	1	1
1	0	1
0	1	1
0	0	0

Semantic Rule: Disjunction
If ϕ and ψ are formulas, then $[\![\phi \vee \psi]\!]^I = 1$ if $[\![\phi]\!]^I = 1$ or $[\![\psi]\!]^I = 1$ (or both), and 0 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:

(15) 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 would be 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 in certain contexts. 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.

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. In this sense, these connectives are TRUTH-FUNCTIONAL. Thanks to this, 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	$[P \wedge Q]$	$\neg[P \wedge Q]$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	1

(The brackets [ ] are crucial here, as they show that we are applying negation to the conjunction of P and Q, rather than P.) Notice that 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.

With these tools, we can prove equivalence and entailment relations between formulas. To do so, construct a truth table with columns for both formulas, and observe how the two columns relate. If two formulas have the same truth values under any possible interpretation (i.e., in every row of the truth table), they are EQUIVALENT. 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: the case where it's true (the first row of truth values) and the case where it's false (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 variables, so there are four cases to consider, as each variable might be either true or false. 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 0s and 1s in the two columns is the same.

P	Q	$[P \wedge Q]$	$\neg[P \wedge Q]$	$\neg P$	$\neg Q$	$[\neg P \vee \neg Q]$
1	1	1	0	0	0	0
1	0	0	1	0	1	1
0	1	0	1	1	0	1
0	0	0	1	1	1	1

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

Exercise 6. De Morgan's Law II: Show that $\neg[P \vee Q]$ is equivalent to $[\neg P \wedge \neg Q]$, using a truth table. 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	$[P \wedge Q]$
1	1	1
1	0	0
0	1	0
0	0	0

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	$[P \vee Q]$	$\neg[P \vee Q]$	$[P \wedge Q]$	$\neg[P \wedge Q]$
1	1				
1	0				
0	1				
0	0				

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

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

(16) Geordi didn't consult Troi and Worf.

It can mean either of the following:

(17) 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 explained based on the relative scope of negation and conjunction. Assume the following basic translations of English sentences into propositional letters:

P = *Geordi consulted Troi.* R = *Geordi consulted Worf.*

The two readings can then be represented as:

(18) a. $\neg[P \wedge R]$
b. $[\neg P \wedge \neg R]$

The two readings are not equivalent. However, the second reading is equivalent to $\neg[P \vee R]$, which could explain 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 (18) captures the reading in (17a)?

(b) Which corresponds to the reading in (17b)?

A note concerning brackets: In general, the outer square brackets with binary connectives are always there according to the offi-

cial 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 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.

Exercise 10. Above, we argued 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 $\neg[P \text{ XOR } Q]$. Where do they differ? Which analysis captures the intuition that *Kim did not invite Pat or Sandy* means 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'). 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 → (sometimes also ⊃).

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

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

(Terminological note: In a conditional sentence of the form 'if A then B', A is called the ANTECEDENT and B 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 (19). 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. The only kind of situation that could falsify the claim is the second one, where the antecedent is true and the consequent is false.

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	$[P \to Q]$
1	1	1
1	0	0
0	1	1
0	0	1

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* in the circumstance where 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. In English, at least, conditional sentences are used to express regularities, so one might reasonably argue that they cannot be judged as true or false in a single situation. In order to capture the meaning of English conditionals, we would need to talk about multiple possible worlds and not just the actual world.[3] But if we are forced to treat the semantics of conditionals as a function of the truth of the antecedent and consequent, the material conditional as we have defined it comes closest to doing the job. With it, we can validate Modus Ponens and invalidate Denying the Antecedent, for example.

Exercise 11. Specify the syntactic and semantic rules for the material conditional.

Exercise 12. Fun fact: $[P \to Q]$ is equivalent to $[\neg P \vee Q]$. Show this by filling in the following truth table.

[3] 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 13. 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	$[P \to Q]$	$\neg Q$	$\neg P$
1	1			
1	0			
0	1			
0	0			

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 14. Recall that Denying the Antecedent has the form:

Premise 1:	$P \rightarrow Q$
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 conditional sentences hold in both directions. 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 conditional sentences do hold in both directions:

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

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

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

The semantics of *if and only if* is captured by the last propositional logic connective we will introduce here: the BICONDITIONAL, written $\leftrightarrow$. In order for $[P \leftrightarrow Q]$ to be true, P and Q must both have the same truth value — either both true or both false. Its truth table looks like this:

P	Q	$[P \leftrightarrow Q]$
1	1	1
1	0	0
0	1	0
0	0	1

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.

Exercise 15. Specify the syntactic and semantic rules for the biconditional.

3.2.4 Equivalence, contradiction and tautology

As mentioned above, if two formulas are true under exactly the same circumstances, 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	$\neg P$	$\neg\neg P$
1	0	1
0	1	0

Exercise 16. Using truth tables, check whether the following pairs of formulas are equivalent.

(a) $[P \vee Q]$; $\neg[\neg P \wedge \neg Q]$

(b) $[P \to Q]$; $[\neg P \vee Q]$

(c) $\neg[P \wedge Q]$; $[\neg P \vee \neg Q]$

(d) $[P \vee \neg Q]$; $\neg[P \wedge \neg Q]$

(e) $[P \to Q]$; $[\neg Q \to \neg P]$

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

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

P	$\neg P$
1	0
0	1

Another contradictory pair is $[P \to Q]$ and $[P \wedge \neg Q]$.

P	Q	$[P \to Q]$	$\neg Q$	$[P \wedge \neg Q]$
1	1	1	0	0
1	0	0	1	1
0	1	1	0	0
0	0	1	1	0

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 1s and 0s 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; otherwise, it is inconsistent. Here is a tautology: $P \vee \neg P$ (e.g. It is raining or it is not raining):

P	$\neg P$	$P \vee \neg P$
1	0	1
0	1	1

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	$\neg P$	$\neg\neg P$	$[P \leftrightarrow \neg\neg P]$
1	0	1	1
0	1	0	1

Exercise 17. Which of the following are tautologies?

(a) $[P \vee Q]$

(b) $[[P \to Q] \vee [Q \to P]]$

(c) $[[P \to Q] \leftrightarrow [\neg Q \vee \neg P]]$

(d) $[[[P \vee Q] \to R] \leftrightarrow [[P \to Q] \wedge [P \to Q]]]$

Support your answer with truth tables.

3.3 Summary: Propositional logic

To summarize what we have covered so far, let us define 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 $\neg\phi$ is a formula.
   - **Binary connectives**: If ϕ and ψ are formulas, then so are:
     - $[\phi \wedge \psi]$ — 'ϕ and ψ'
     - $[\phi \vee \psi]$ — 'ϕ or ψ'
     - $[\phi \rightarrow \psi]$ — 'if ϕ then ψ'
     - $[\phi \leftrightarrow \psi]$ — 'ϕ if and only if ψ'

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

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

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

1. **Propositional letters**

   - If ϕ is any propositional letter, then

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

2. **Complex formulas**

   - **Unary connective** If ϕ is a formula, then $[\![\neg\phi]\!]^I = 1$ if $[\![\phi]\!]^I = 0$, and 0 otherwise.

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

   - $[\![\phi \wedge \psi]\!]^I = 1$ if $[\![\phi]\!]^I = 1$ and $[\![\psi]\!]^I = 1$, and 0 otherwise.
   - $[\![\phi \vee \psi]\!]^I = 1$ if $[\![\phi]\!]^I = 1$ or $[\![\psi]\!]^I = 1$ (or both), and 0 otherwise.
   - (Semantic rules for $\rightarrow$ and $\leftrightarrow$ were left as exercises.)

4 Predicate logic

4.1 From propositional logic to predicate logic

In natural languages, sentences may or may not consist of other sentences. For example, the English sentence *Abelard is happy and Eloise is sad* contains two sub-sentences, so to speak, *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*. More generally, in propositional logic, an atomic formula consists of just one propositional letter, but natural sentences that atomic formulas

[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.

represent can consist of multiple words.

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, or more specifically FIRST-ORDER 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 of the language with no internal structure: names like a (for Abelard) and e (for Eloise), predicates like Happy and Loves, and functions like 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 4.3.

4.1.1 Individual constants

Predicate logic is a language that allows us to reason about a given DOMAIN consisting of individual objects. Individual objects are named by INDIVIDUAL CONSTANTS or NAMES. In this book, we adopt the convention that individual constants start with a lowercase letter, and 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 (a bit like a noun phrase 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 additional tasks. One of them is to map individual constants to individuals. As mentioned above, a set of individuals that are available for this purpose is called a DOMAIN, which we will write as D. A pair $\langle D, I \rangle$ is called a MODEL for predicate logic, and we will write it as M.

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 *Agnetha*, *Björn*, *Benny*, and *Anni-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: $\mathsf{ag}, \mathsf{bj}, \mathsf{be}, \mathsf{fr}$. Let us also assume that our interpretation function maps these constants to the appropriate band members. For this purpose, we will define a model $M_0 = \langle D_0, I_0 \rangle$ and give the following partial definition of I_0:

(1) a. $I_0(\mathsf{ag})$ = Agnetha
b. $I_0(\mathsf{bj})$ = Björn
c. $I_0(\mathsf{be})$ = Benny
d. $I_0(\mathsf{fr})$ = Frida

Unlike names in English such as *Björn*, individual constants are not ambiguous; they pick out exactly one object. However, not every individual object in a given model needs to have a corresponding individual constant in a given language; we might have any number of objects in the model's domain (say, Benny's nose) that are not named by any individual constant in the language. There is an important distinction between the objects themselves, which are not part of the formal language, and the names for those objects, which are.

Just as in propositional logic, we will define a DENOTATION FUNCTION that coincides with I on individual constants and ex-

tends it to formulas 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. $[\![\text{ag}]\!]^{M_0}$ = Agnetha
b. $[\![\text{bj}]\!]^{M_0}$ = Björn
c. $[\![\text{be}]\!]^{M_0}$ = Benny
d. $[\![\text{fr}]\!]^{M_0}$ = Frida

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 metavariable. 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. We will see other examples of metavariables later on.

This rule refers to 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 relative to a given model. LOGICAL CONSTANTS are things like $\wedge$, whose denotation does not even vary from model to model; the denotations of NON-LOGICAL CONSTANTS, on the other hand, depend on the model. Later, we will introduce variables, whose denotation does vary even once a model has been specified.

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):

> There would be little chance of confusion on this matter were it not for the fact that we are communicating with reader by means of the printed page, and so we could not put [on the right-hand side of the equation in (2a)] the [pop singer Agnetha herself] but rather have let [her] be represented by [her] conventional [name] in English ... The point is worth belaboring since it is central to the program of truth conditional semantics ... that a connection is made between language and extra-linguistic reality, i.e. "the world." (The sanitizing quotes here are prompted by the fact that we will eventually want to consider not only the world in which we live as it actually is but also the world as it was, as it will be, as it might have been, etc. i.e., other "possible worlds".)

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 using the printed page.

In Chapter 6, we will define a system that relates English expressions to logical expressions like this, and thereby indirectly associates English expressions with denotations in the domain of a 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* $\rightsquigarrow$ bj (English to logic)
- $[\![\text{bj}]\!]^{M_0}$ = Björn (logic to denotation)

We follow the convention of italicizing object language expressions here and throughout.

We say that *Björn* TRANSLATES TO bj and that bj (and, indirectly, *Björn*) DENOTES Björn. And similarly for other expressions.

For example, we will map adjectives to appropriate predicates (to be introduced in the next section) and interpret these predicates as appropriate sets and relations.

- *female* $\rightsquigarrow$ Female (English to logic)
- $[\![\text{Female}]\!]^{M_0} = \{\text{Agnetha}, \text{Frida}\}$ (logic to denotation)

Again, the style of doing semantics we are adopting here is called INDIRECT INTERPRETATION. There is another style, known as DIRECT INTERPRETATION, where the mapping is carried out in one fell swoop:

- $[\![\textit{Björn}]\!] = \text{Björn}$ (English to denotation)
- $[\![\textit{female}]\!] = \{\text{Agnetha}, \text{Frida}\}$ (English to denotation)

Both styles are common in semantic theory. The direct interpretation style is used in the classic Heim & Kratzer (1998) textbook. However, indirect interpretation carries a number of practical advantages, as discussed in Chapter 1.

4.1.2 Predication

4.1.2.1 Syntax of predication

True to its name, predicate logic has PREDICATES, along with individual constants. These are expressions that pick out sets of objects. A predicate is a symbol (a string of letters, such as like Female, Swedish or Sings) that denotes a set (such as the set of female or Swedish or singing individuals). Sometimes, the term PREDICATE SYMBOL is used to refer to the symbol, and the term PREDICATE is then used just for the set that it denotes. In this textbook, we will always capitalize the first letter of a predicate symbol.

Each of the examples just given is a UNARY PREDICATE, because it applies to one term (called its ARGUMENT) to produce a statement that can be true or false. For example:

(3) Swedish(ag)

will be interpreted as saying that Agnetha is Swedish. We say that the predicate Swedish HOLDS OF or APPLIES TO ag. The complex consisting of the predicate with its argument is called a FORMULA. Like in propositional logic, formulas can be true or false. We will encounter other types of formulas later on.

A BINARY PREDICATE denotes a relation between two individuals, and therefore takes two arguments. As an example of a binary predicate, we will use Loves, and we will assume that it denotes the relation (the set of pairs) that contains a given pair of two individuals just in case the first loves the second. A binary predicate takes two arguments:

(4) Loves(ag,bj)

We say that the predicate Loves HOLDS OF, APPLIES TO, or RELATES ag and bj.

The number of arguments that a predicate takes is its ARITY, also called VALENCE or ADICITY. Unary predicates take one argument, 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, as holding of three objects x, y and z, 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. In general, predicates combine with the appropriate number of arguments to form ATOMIC FORMULAS. For example,

Singer(ag)

is an atomic formula, expressing the fact that Agnetha is a singer. Here a unary predicate Singer combines with a single argument, enclosed in parentheses, to form an atomic formula. A binary

predicate combines with two arguments, enclosed in parentheses, to form an atomic formula. Thus:

$$\mathsf{Loves}(\mathsf{ag}, \mathsf{bj})$$

is also an atomic formula, expressing the fact that Agnetha loves Björn. In general:

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.)

It follows from the syntactic rule of predication that 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 one 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 arguments.

(6) a. $\mathsf{Excited1}(\mathsf{ag})$
b. $\mathsf{Excited2}(\mathsf{ag}, \mathsf{be})$

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:

- ag and bj 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 propositions corresponding to the following formulas were true:

(7) a. Loves(ag,bj)
b. Loves(bj,ag)

(8) a. Loves(be,fr)
b. Loves(fr,be)

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 $M_{\text{NOW}} = \langle D_0, I_{\text{NOW}} \rangle$.

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

(10) $I_{\text{NOW}}(\text{Loves}) = \{\}$

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 model the denotation of the binary predicate Loves as a binary relation, that is, a set of ordered pairs. A ternary predicate will denote a ternary relation, etc.

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(ag) is a truth value:

$$[\![\text{Singer(ag)}]\!]^M = 1 \text{ if } [\![\text{ag}]\!]^M \in [\![\text{Singer}]\!]^M, \text{ and 0 otherwise.}$$

The denotations of unary predicates can differ across models. For example, in some models, Frida is a singer, while in other models, she stays off the microphone and sticks to the synthesizer. Whether or not she is in the extension of the unary predicate Singer, and hence, whether the formula Singer(fr) is true, depends on whether the denotation of fr (namely Frida) is in the set

denoted by 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:

$$[\![\pi(\alpha)]\!]^M = 1 \text{ if } [\![\alpha]\!] \in [\![\pi]\!]^M, \text{ and 0 otherwise.}$$

This can be read, "the semantic value of *pi* applied to *alpha* (with respect to model M) is 1, if the semantic value of *alpha* (with respect to M) is an element of the semantic value of *pi* (with respect to M), and 0 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 Loves(ag, bj) is true relative to M_{THEN} and false relative to M_{NOW}.

(11) a. $[\![\text{Loves(ag, bj)}]\!]^{M_{\text{THEN}}} = 1$
because $\langle \text{Agnetha, Björn} \rangle \in [\![\text{Loves}]\!]^{M_{\text{THEN}}}$

b. $[\![\text{Loves(ag, bj)}]\!]^{M_{\text{NOW}}} = 0$
because $\langle \text{Agnetha, Björn} \rangle \notin [\![\text{Loves}]\!]^{M_{\text{NOW}}}$

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

$$[\![\pi(\alpha, \beta)]\!]^M = 1 \text{ if } \langle [\![\alpha]\!]^M, [\![\beta]\!]^M \rangle \in [\![\pi]\!]^M, \text{ and 0 otherwise.}$$

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:

$$[\![\pi(\alpha_1, ...\alpha_n)]\!]^M = 1 \text{ if } \langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M \rangle \in [\![\pi]\!]^M, \text{ and 0 otherwise.}$$

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, 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 contains Abelard and Eloise:

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

and I_3 is defined as follows:

$$I = \begin{bmatrix} \text{a} & \rightarrow & \text{Abelard} \\ \text{e} & \rightarrow & \text{Eloise} \\ \text{Female} & \rightarrow & \{\text{Eloise}\} \\ \text{Scholar} & \rightarrow & \{\text{Abelard, Eloise}\} \\ \text{Loves} & \rightarrow & \{\langle\text{Abelard,Eloise}\rangle, \langle\text{Eloise,Abelard}\rangle, \langle\text{Eloise,Eloise}\rangle\} \\ \text{Teacher} & \rightarrow & \{\langle\text{Abelard,Eloise}\rangle\} \end{bmatrix}$$

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)

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 FUNCTIONS. Within the confines of monogamy, an example of a function would be spouseOf, which, combined with the name of an individual, denotes the spouse of that individual. For example,

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

could be used to denote the spouse of Benny. It does not *say* something *about* Benny, like a predicate does, and the combination of function with its argument does not make a truth-evaluable *claim*, as in the case of a predicate. Rather, this expression denotes a particular individual.

Syntactically, names and complex terms are identical. They can appear in all the same positions. For instance, complex terms can serve as the first argument to a binary relation:

$$\mathsf{Loves}(\mathsf{spouseOf}(\mathsf{be}), \mathsf{be})$$

This formula says that Benny's spouse loves him (Benny).

Predicates and functions are easy to confuse with each other, because they both take arguments in parentheses. To distinguish between them, this textbook uses the following conventions: predicates are written with uppercase letters, and functions are written with lowercase letters and end in Of. This way, all terms (both individual constants and complex terms formed with functions) start with lowercase letters. Any sequence of numbers or letters or underscores may follow the initial letter, but no spaces.

Functions, like predicates, have a particular arity. The spouseOf function has arity 1 (i.e, it is a UNARY FUNCTION). An example of a function with arity 2 might be tallerOf, which takes two arguments and returns whichever one is taller. Something like:

$$\mathsf{tallerOf}(\mathsf{ag}, \mathsf{fr})$$

would denote Frida, if she is the taller of the two. 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.

Function symbols denote functions (of the kind discussed in the previous chapter, namely, relations of a special kind); this denotation is specified by the interpretation function I. Functions are considered non-logical constants; therefore, their denotation is derived from I according to the same rule as individual constants and predicates. However, functions combine with their arguments in a slightly different manner from the way predicates 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. For example, suppose that in model M, spouseOf denotes a function that gives Frida when given the individual Benny as an argument. Then spouseOf(be) denotes Frida, i.e., $[\![\text{spouseOf(be)}]\!]^M$ = 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 with respect to model M is equal to the semantic value of gamma with respect to M applied to the semantic value of alpha with respect to M." Note that we are using parentheses both

in the object language and the meta-language here. The parentheses in the object language connect the function symbol to a term. The parentheses in the object language signify the application of the denoted function to the actual individual denoted by the term.

Binary functions take ordered pairs as arguments. For example, the following expression:

$$\mathsf{tallerOf}(\mathsf{ag},\mathsf{fr})$$

denotes the result of applying the function denoted by tallerOf to the ordered pair ⟨Agnetha,Frida⟩. 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 $\alpha_1,\ldots,\alpha_n$ is a sequence of n terms, then:

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

Exercise 4. Which of the following are well-formed formulas? Assume Happy is a predicate that takes one argument, spouseOf is a unary function, and tallerOf is a binary function.

(a) Happy(spouseOf(ag))

(b) Happy(spouseOf(ag,be))

(c) $\mathsf{Happy}(\mathsf{tallerOf}(\mathsf{ag},\mathsf{be}))$

(d) $\mathsf{Happy}(\mathsf{spouseOf}(\mathsf{ag}),\mathsf{spouseOf}(\mathsf{be}))$

4.1.4 Equality

Atomic formulas can be formed in two ways. Predication, as we have just seen, is one of them. Another way to produce an atomic formula is by joining two terms with an 'equals' symbol, like so:

$$\mathsf{ag} = \mathsf{be}$$

$$\mathsf{spouseOf}(\mathsf{ag}) = \mathsf{bj}$$

$$\mathsf{fr} = \mathsf{tallerOf}(\mathsf{fr},\mathsf{ag})$$

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

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

The corresponding semantic rule is as you might expect:

Semantic Rule: Equality
If α and β are terms, then $[\![\alpha = \beta]\!]^M = 1$ if $[\![\alpha]\!]^M = [\![\beta]\!]^M$, and 0 otherwise.

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.

Exercise 5. Which of the following is well-formed?

(a) $\mathsf{spouseOf(ag)} = \mathsf{spouseOf(be)}$

(b) $\mathsf{spouseOf(ag)} \leftrightarrow \mathsf{spouseOf(be)}$

Summary

To summarize, we have the following types of basic expressions, which are all non-logical constant symbols in our language.

CATEGORY	EXAMPLE
individual constants	ag
unary predicates	Singer
binary predicates	Loves
function symbols	spouseOf

The interpretation function of a model determines the semantic values of these constant symbols. Atomic formulas can be produced either through predication or equality.

4.2 Summary: L_0

To summarize what we have covered so far, let us define a simple language called L_0. 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.

4.2.1 Syntax of L_0

1. **Basic Expressions**

- **Individual constants:** ag, be, bj, fr
- **Function symbols**
  - Unary: spouseOf, selfOf
  - Binary: tallerOf
- **Predicates**
  - Unary: Swedish, Singer
  - Binary: Loves

2. **Terms**
   - **Names:** Every individual constant is a term.
   - **Complex terms:** If π is a function symbol of arity n, and $\alpha_1, ..., \alpha_n$ are terms, then $\pi(\alpha_1, ..., \alpha_n)$ is a term.[2]
3. **Atomic formulas**
   - **Predication:** If π is a predicate symbol of arity n and $\alpha_1, ...\alpha_n$ are terms, then $\pi(\alpha_1, ...\alpha_n)$ is a formula.[3]
   - **Equality:** If α and β are terms, then $\alpha = \beta$ is a 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:

[2] 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.

[3] Special cases:
- If π is a unary predicate symbol and α is a term, then $\pi(\alpha)$ is a formula.
- If π is a binary predicate symbol and α and β are terms, then $\pi(\alpha, \beta)$ is a formula.

- **Conjunction:** $[\phi \wedge \psi]$ 'ϕ and ψ'
- **Disjunction:** $[\phi \vee \psi]$ 'ϕ or ψ'
- **Conditional:** $[\phi \rightarrow \psi]$ 'if ϕ then ψ'
- **Biconditional:** $[\phi \leftrightarrow \psi]$ 'ϕ if and only if ψ'

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

4.2.2 Semantics of L_0

The denotation of an expression α relative to a model M is written $[\![\alpha]\!]^M$. A model $M = \langle D, I \rangle$ determines a domain of individuals D and an interpretation function I.

1. **Basic expressions**
   - If α is a non-logical constant (individual constant, predicate symbol, or function symbol), then $[\![\alpha]\!]^M = \mathrm{I}(\alpha)$.
2. **Complex terms**
   - If π is a function of arity n, and $\alpha_1, ..., \alpha_n$ is a sequence of n terms, then $[\![\pi(\alpha_1, ..., \alpha_n)]\!]^M = [\![\pi]\!]^M(\langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M \rangle)$[4]
3. **Atomic formulas**
   - **Predication**: If π is a predicate of arity n, and $\alpha_1, ..., \alpha_n$ is a sequence of n terms, then $[\![\pi(\alpha_1, ..., \alpha_n)]\!]^M = 1$ if $\langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M \rangle \in [\![\pi]\!]^M$, and 0 otherwise.[5]

[4] Special cases:
- When π is a function of arity 1, then $[\![\pi(\alpha)]\!] = [\![\pi]\!]^M([\![\alpha]\!]^M)$.
- When π is a function of arity 2, then $[\![\pi(\alpha, \beta)]\!] = [\![\pi]\!]^M(\langle [\![\alpha]\!]^M, [\![\beta]\!]^M \rangle)$.

[5] Special cases:

- **Equality:** If α and β are terms, then $[\![\alpha = \beta]\!]^M = 1$ if $[\![\alpha]\!]^M = [\![\beta]\!]^M$, and 0 otherwise.

4. **Negation**
   - $[\![\neg\phi]\!]^M = 1$ if $[\![\phi]\!]^M = 0$, and 0 otherwise.
5. **Binary connectives**
   - $[\![\phi \wedge \psi]\!]^M = 1$ if $[\![\phi]\!]^M = 1$ and $[\![\psi]\!]^M = 1$, and 0 otherwise.
   - $[\![\phi \vee \psi]\!]^M = 1$ if $[\![\phi]\!]^M = 1$ or $[\![\psi]\!]^M = 1$, and 0 otherwise.
   - (Semantic rules for $\rightarrow$ and $\leftrightarrow$ were left as exercises.)

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. Fill in the missing values in the interpretation function, according to what you think they should be based on the constant symbol:

$$I_4 = \left[\begin{array}{lll} \text{a} & \rightarrow & \text{Abelard} \\ \text{e} & \rightarrow & \text{Eloise} \\ \text{Female} & \rightarrow & \{\text{Eloise}\} \\ \text{Male} & \rightarrow & \\ \text{Scholar} & \rightarrow & \{\text{Abelard, Eloise}\} \\ \text{Loves} & \rightarrow & \{\langle\text{Abelard,Eloise}\rangle, \langle\text{Eloise,Abelard}\rangle, \langle\text{Eloise,Eloise}\rangle\} \\ \text{Younger} & \rightarrow & \{\langle\text{Eloise,Abelard}\rangle\} \\ \text{spouseOf} & \rightarrow & \{\langle\text{Abelard,Eloise}\rangle, \langle\text{Eloise,Abelard}\rangle\} \\ \text{selfOf} & \rightarrow & \end{array}\right]$$

– If π is a unary predicate and α is a term, then $[\![\pi(\alpha)]\!]^M = 1$ iff $[\![\alpha]\!]^M \in [\![\pi]\!]^M$.

– If π is a binary predicate and α and β are terms, then $[\![\pi(\alpha,\beta)]\!]^M = 1$ iff $\langle [\![\alpha]\!]^M, [\![\beta]\!]^M \rangle \in [\![\pi]\!]^M$.

Exercise 7. Fill in the following table.

	Term or formula?	$[\![\cdot]\!]^{M_4}$	**Semantic Rule(s)**
spouseOf(a)	term	Eloise	Basic expressions
Female(a)	formula	0	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. ∴ 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 Shiela 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 Brother as your translation for *brother* and 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). 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:

$$\begin{array}{l}[\phi \wedge \psi] \\ \therefore \phi\end{array}$$

(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.)

4.3 Quantification

Consider again this syllogism:

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

Construing teaching as a binary relation that holds between teachers and their students, the conclusion of this syllogism is true in any model where Aristotle stands in the teaching relation to an entity that is a king. How can we express this formally? If we had names for all of the kings in the model, then we could express this using the tools we have by saying something along the lines of, "Aristotle taught King 1 or Aristotle taught King 2 or ..." and so on for all of the kings. But this is quite inconvenient. 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. The condition that the object should satisfy may be written as follows:

$$[\mathsf{Taught}(\mathsf{aristotle}, x) \wedge \mathsf{King}(x)]$$

This is a well-formed formula of first-order logic, but it does not make a claim; it just describes a condition that some object x might or might not satisfy. This is because the variable x is not BOUND by any quantifier (so it is FREE). To make the claim that there is some object x that satisfies this condition, we may use the EXISTENTIAL QUANTIFIER, $\exists$.

$$\exists x[\mathsf{Taught}(\mathsf{aristotle}, x) \wedge \mathsf{King}(x)]$$

This can be read, "There exists an x such that Aristotle taught x and x is a king." And this formula will be true in any model where there is a king that Aristotle taught.

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 above, we would have expressed the claim that *everything* satisfies the condition. 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 athlete got an A*. We can represent this as follows:

$$\forall x[\mathsf{Athlete}(x) \rightarrow \mathsf{ReceivedGrade}(x, \mathsf{a})]$$

This can be read, "For all x, if x is an athlete, then the grade x received was an A." 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:

$$\forall x[\mathsf{Athlete}(x) \wedge \mathsf{ReceivedGrade}(x, \mathsf{a})]$$

This says, "For all x, x is an athlete and x received an A" – in other words, "Everything is an athlete and everything received an A."

Let us take some time to reflect on why the universally quantified formula with the material conditional above expresses the *every* claim, that every athlete got an A. 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 satisifies the condition:

$$[\mathsf{Athlete}(x) \to \mathsf{ReceivedGrade}(x, \mathsf{a})]$$

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 an athlete, or the value of x is not an athlete. Consider a value for x who is not an athlete. For this value of x, the condition

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

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 an athlete vacuously satisfies $[\mathsf{Athlete}(x) \to \mathsf{ReceivedGrade}(x, \mathsf{a})]$. The only kind of value for x that could fail to satisfy this condition would be an athlete that did not get an A. 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 athletes that did not get an A, then the formula is true. And this is exactly what *Every athlete got an A* says.

Now consider the following formula:

$$\forall x[\mathsf{Linguist}(x) \to \exists y[\mathsf{Philosopher}(y) \wedge \mathsf{Admires}(x, y)]]$$

If we were to read this aloud, symbol for symbol, we would say, "For every x, if x is a linguist, then there exists a y such that y is a philosopher and x admires y." A more natural way of putting this would be "Every linguist admires a philosopher." But "Every linguist admires a philosopher" is actually ambiguous. It could mean two things:

1. For every linguist, there is some philosopher that the linguist admires (possibly a different philosopher for every linguist).

2. There is one lucky philosopher such that every linguist admires that philosopher.

The latter reading could be rendered logically as follows:

$$\exists y[\mathsf{Philosopher}(y) \wedge \forall x[\mathsf{Linguist}(x) \rightarrow \mathsf{Admires}(x, y)]]$$

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:

$$\forall x.\neg\mathsf{Happy}(x)$$

$$\neg\forall x.\mathsf{Happy}(x)$$

The one where the universal quantifier takes wide scope over negation says, "For every x, it is not the case that x is happy." The one 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 implies that nobody is happy. The second one implies 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 L_1:

(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 L_1. In some cases, there may be quantifier scope ambiguity; in that case, give a representation in L_1 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 defining the syntax of this language formally. We will allow an infinite number of variables v_0, v_1, v_2, ... all of type e, but use the following shorthands:

- x is v_0
- y is v_1
- z is v_2

We will also add new formation rules for the universal quantifier $\forall$, and the existential quantifier $\exists$.

Syntactic rule for L_1: Quantification
Given any variable u, if ϕ is a formula, then

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

is a formula, and so is

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

For example, $\forall x.\mathsf{Happy}(x)$ is a valid formula according to these rules. As an abbreviatory shorthand, we may drop the dot after the variable when it is immediately followed by a bracket, e.g.

$\forall x[\mathsf{Happy}(x) \rightarrow \mathsf{Friendly}(x)]$. In a formula of the form $\forall u.\phi$ or $\exists u.\phi$, ϕ is called the SCOPE of the quantifier.

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,

$$\forall x.\mathsf{Happy}(x)$$

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

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

is true. Likewise, informally,

$$\exists x.\mathsf{Happy}(x)$$

is true iff we can find *some* individual to assign to x that makes $\mathsf{Happy}(x)$ true.

As we have seen, a formula can in principle have multiple quantifiers. For example:

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

This says, 'everything that is happy likes something.' Whether or not it is true, it contains two variables and two quantifiers. The outermost formula is true if every individual in the domain is an x such that:

$$[\mathsf{Happy}(x) \rightarrow \exists y.\mathsf{Likes}(x,y)]$$

In order to evaluate whether this holds for a given x that is happy, we will need to determine whether there is a y that x likes. So we will need to hold x constant 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.

An assignment function is a function that specifies for each variable, how that variable is to be interpreted. 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 (v_n for all n).

In order to interpret an expression like $\mathsf{Happy}(x)$, 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 x with respect to model M and assignment function g*, written:

$$[\![x]\!]^{M,g}$$

is simply whatever g maps x to. We can express this more formally as follows:

Semantic rule for L_1: Variables
$[\![x]\!]^{M,g} = g(x)$

For example, $[\![x]\!]^{M,g_1} = g_1(x) =$ Agnetha, and $[\![x]\!]^{M,g_2} = g_2(x) =$ Benny for any 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. 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 model M and 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 by considering a variant of g on which the variable x is assigned to some happy individual.

Let us use the expression

$$g[u \mapsto k]$$

to describe an assignment function that is exactly like g with the possible exception that $g(u) = k$. For example,

$$g[u \mapsto \text{Agnetha Fältskog}]$$

is an assignment function that is exactly like g with the possible exception that the value of $g(u)$ is Agnetha Fältskog. Thus k is a symbol of our meta-language that stands for an individual in the domain. If g already maps u to k then, $g[u \mapsto k]$ is the same as g. We call this a u-VARIANT OF g. This 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:

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

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

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

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 $\exists x.\mathsf{Happy}(x)$:

$$[\![\exists x.\mathsf{Happy}(x)]\!]^{M,g} = 1 \text{ iff there is an individual } k \in D \text{ such that: } [\![\mathsf{Happy}(x)]\!]^{M,g[x\mapsto k]} = 1.$$

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
$[\![\exists x.\phi]\!]^{M,g} = 1$ iff there is an individual $k \in D$ such that:

$$[\![\phi]\!]^{M,g[x\mapsto k]} = 1$$

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
$[\![\forall v\,.\,\phi]\!]^{M,g} = 1$ iff for all individuals $k \in D$:

$$[\![\phi]\!]^{M,g[v \mapsto k]} = 1$$

4.3.1 Syntax of L_1

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:** ag, bj, be, fr, ...
   - **Individual variables:** v_n for every natural number n
   - **Function symbols**
     - Unary: spouseOf, ...
     - Binary: tallerOf, ...
   - **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 $\alpha_1, ..., \alpha_n$ are terms, then $\pi(\alpha_1, ..., \alpha_n)$ is a term.[6]

[6] 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.

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.[7]
   - **Equality**
     If α and β are terms, then $\alpha = \beta$ is an atomic formula.

4. **Negation**

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

5. **Binary connectives**
   If ϕ is a formula and ψ is a formula, then so are:

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

6. **Quantifiers**
   If u is a variable and ϕ is a formula, then both of the following are formulas:

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

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.

[7] 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.

- 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. A formula containing 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.

Before moving on to the semantics, let us establish some abbreviatory conventions: We want to avoid unnecessary clutter in our representations, so we allow brackets to be dropped when it is independently clear what the scope of a quantifier is, and we also allow the outermost brackets of an expression to be dropped. For example, instead of:

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

we can write:

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

because it is clear that the scope of the existential quantifier does not extend any farther to the right than it does. Furthermore, when reading a formula, you may assume that the scope of a binder (e.g. $\forall x$ or $\exists x$) extends as far to the right as possible. So, for example, $\forall x[P(x) \wedge Q(x)]$ can be rewritten as $\forall x.P(x) \wedge Q(x)$, interpreted in such a way that the universal quantifier takes scope over the conjunction, rather than as the conjunction of $\forall x.P(x)$ and $Q(x)$. (As a heuristic, you may think of the dot as a "wall" that

forms the left edge of a constituent, which continues until you find an unbalanced right bracket or the end of the expression.) However, we will typically retain brackets around conjunctions, disjunctions, and implications.

We retain all of the abbreviatory conventions from above in order to avoid unnecessary clutter in our formulas. Furthermore, we can drop the dot between two quantificational binders in a row. Thus instead of:

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

we can write:

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

This convention is specific to our textbook, and there is no single standard in the field. In the Lambda Calculator, on its default setting, dots are *always* optional.

4.3.2 Semantics of L_1

Now for the semantics of L_1. 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_1 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:[8]

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

3. **Atomic formulas**

   - **Predication**
     If π is a predicate of arity n and $\alpha_1, ..., \alpha_n$ is a sequence of terms, then: $[\![\pi(\alpha_1, ...\alpha_n)]\!]^M = 1$ if $\langle [\![\alpha_1]\!]^M, ..., [\![\alpha_n]\!]^M\rangle \in [\![\pi]\!]^M$, and 0 otherwise.
   - **Equality**
     If α and β are terms, then

$$[\![\alpha = \beta]\!]^{M,g} = 1 \text{ if } [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g},$$

     and 0 otherwise.

4. **Negation**

   - $[\![\neg\phi]\!]^{M,g} = 1$ if $[\![\phi]\!]^{M,g} = 0$, and 0 otherwise.

5. **Binary Connectives**

   - $[\![\phi \wedge \psi]\!]^{M,g} = 1$ if $[\![\phi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$, and 0 otherwise.
   - $[\![\phi \vee \psi]\!]^{M,g} = 1$ if $[\![\phi]\!]^{M,g} = 1$ or $[\![\psi]\!]^{M,g} = 1$, and 0 otherwise.

[8] Special cases:

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

$$[\![\pi(\alpha)]\!] = [\![\pi]\!]^{M,g}([\![\alpha]\!]^{M,g}).$$

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

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

- (Semantic rules for $\rightarrow$ and $\leftrightarrow$ were left as exercises.)

6. **Quantification**

   - $[\![\forall v.\phi]\!]^{M,g} = 1$ if for all individuals $k \in D$:

     $$[\![\phi]\!]^{M,g[v \mapsto k]} = 1$$

     and 0 otherwise.
   - $[\![\exists v.\phi]\!]^{M,g} = 1$ if there is an individual $k \in D$ such that:

     $$[\![\phi]\!]^{M,g[v \mapsto k]} = 1$$

     and 0 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:

$$\forall x[\mathsf{Linguist}(x) \rightarrow \exists y[\mathsf{Philosopher}(y) \wedge \mathsf{Admires}(y)(x)]]$$

$$\exists y[\mathsf{Philosopher}(y) \wedge \forall x[\mathsf{Linguist}(x) \rightarrow \mathsf{Admires}(y)(x)]]$$

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_1?

(ii) Of the ones that are well formed in L_1, which of the above formulas 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:

$$I_f(\mathsf{Happy}) = \{\text{Benny, Björn, Agnetha}\}$$

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_1 semantic interpretation rule for variables.

(b) What is $[\![\mathsf{Happy}]\!]^{M_f, g_{\text{Benny}}}$? Apply the relevant L_1 semantic interpretation rule.

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

(d) Using the rule you identified in your answer to the previous question, explain carefully why $[\![\mathsf{Happy}(x)]\!]^{M_f, g_{\text{Benny}}} = 1$.

Exercise 18. Consider the following four assignment functions.

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

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

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_1. 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) $\mathsf{Teacher}(x, y) \rightarrow \mathsf{Male}(x)$

(s) $\forall y [\mathsf{Teacher}(x, y) \rightarrow \mathsf{Male}(x)]$

(t) $\forall x \forall y [\mathsf{Teacher}(x, y) \rightarrow \mathsf{Male}(x)]$

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 $[\![\forall x\,.\,\mathsf{Loves}(x,\mathsf{f})]\!]^{M,g}$ 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 $[\![\mathsf{Loves}(x,\mathsf{f})]\!]^{M,g_i} \neq [\![\mathsf{Loves}(x,\mathsf{f})]\!]^{M,g_j}$.

Exercise 21. In the run-up to the 2016 U.S. presidential election, Stephen Colbert once joked as follows:

> In a recent CNN poll of New Hampshire Republicans, [Bobby] Jindal got 3% of respondents, tying with Rick Santorum, and falling just short of "No-one" at 4%. Which I say he can use to his advantage: "Jindal 2016: No one is more popular!"

Explain this joke in painstaking detail by giving translations into L_1 of the two interpretations of the proposed slogan that the joke plays on. Feel free to introduce as many non-logical constants as you need.

5 Typed lambda calculus

5.1 Introduction

As you may recall from the introduction, this book explores the idea that semantic composition involves a kind of saturation that can be modelled using functions. Suppose you have a syntactic phrase consisting of two sub-phrases, such as a sentence made up of a subject and a verb phrase, or a verb phrase made up of a transitive verb and its object. In order for the meanings 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 for describing functions. Doing will entable us to model semantic composition quite elegantly.

Consider the sentence *John loves Mary*, which might be translated into L_1 as:

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

Some parts of the sentence *John loves Mary* can be straightforwardly mapped into expressions in L_1, 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 rela-

tion (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:

$$\mathsf{Loves}(\_\_\_\_, \mathsf{m})$$

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, denoted with 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:

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

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—more or less what *loves Mary* means.

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

$$\forall x[\_\_\_\_(x)]$$

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

$$\forall x[\mathsf{Temporary}(x)]$$

while *Everything is permanent* would be expressed:

$$\forall x[\mathsf{Permanent}(x)]$$

In predicate logic, variables only range over individuals. But the missing piece is a predicate in the example under consideration. So we will now switch to a language of HIGHER-ORDER LOGIC. This means we can have variables ranging over predicates, which can then be abstracted over. The result looks like this:

$$\lambda P\,.\,\forall x[P(x)]$$

This expression (read 'lambda P dot for all x: P 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. 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 $\mathrm{L}_{\mathrm{Prop}}$ and L_0 had a rather limited set of syntactic categories: terms, predicates of various arities, and formulas. In the language L_λ that we present next, we will have a much richer set of syntactic categories, called TYPES. A type represents the kind of denotation an expression denotes, 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, although there are strict rules as to what counts as a type and what doesn't. We will start with two BASIC TYPES:

$$e$$

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

$$t$$

(the type of truth values) for formulas. From these types we will build up FUNCTION TYPES such as:

$$\langle e, t \rangle$$

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 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, not categories of denotations. However, they are associated with categories of denotations. Each type has an associated 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. An expression of type t is a formula, so its denotation must be either 1 or 0;

$D_t = \{1, 0\}$. 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. 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 $\langle e\rangle$ when they mean e); according to our definition, angle brackets are only introduced for *function types*.

We also lack 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 relation, using a method known as CURRYING the underlying binary relation.[1] For a given binary relation R, LEFT-TO-RIGHT CURRYING produces a function f such that $[f(x)](y) = 1$ if and only if $\langle x, y\rangle \in R$ (where $[f(x)](y)$ denotes the result of first applying f to x, and then applying $f(x)$ to y). Analogously, right-to-left currying produces a function f such that $[f(y)](x) = 1$ if and only if $\langle x, y\rangle \in R$.

An example denotation for a type $\langle e, \langle e, t\rangle\rangle$ expression might

[1] The term 'currying' is named after the logician Haskell Curry. 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]."

be the following function:

$$\left[\begin{array}{lll}
\text{Agnetha} & \to & \left[\begin{array}{lll} \text{Agnetha} & \to & 0 \\ \text{Benny} & \to & 0 \\ \text{Björn} & \to & 0 \\ \text{Frida} & \to & 0 \end{array}\right] \\
\text{Benny} & \to & \left[\begin{array}{lll} \text{Agnetha} & \to & 0 \\ \text{Benny} & \to & 0 \\ \text{Björn} & \to & 1 \\ \text{Frida} & \to & 0 \end{array}\right] \\
\text{Björn} & \to & \left[\begin{array}{lll} \text{Agnetha} & \to & 0 \\ \text{Benny} & \to & 0 \\ \text{Björn} & \to & 1 \\ \text{Frida} & \to & 1 \end{array}\right] \\
\text{Frida} & \to & \left[\begin{array}{lll} \text{Agnetha} & \to & 1 \\ \text{Benny} & \to & 0 \\ \text{Björn} & \to & 0 \\ \text{Frida} & \to & 0 \end{array}\right]
\end{array}\right]$$

Call this function f. Applied to a given argument, say Björn, it gives back another function:

$$f(\text{Björn}) = \left[\begin{array}{lll} \text{Agnetha} & \to & 0 \\ \text{Benny} & \to & 0 \\ \text{Björn} & \to & 1 \\ \text{Frida} & \to & 1 \end{array}\right]$$

This function f is what results when the following relation is right-to-left curried:

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

Again, a pair $\langle x, y\rangle$ is in the relation just in case $f(y)$ applied to x yields 1.

As it turns out, curried relations are precisely what we need in order to give a compositional analysis of sentences in natural languages containing transitive verbs. 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

$$\mathsf{loves}$$

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

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

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

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

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*:

$$\mathsf{Loves}(\mathsf{a},\mathsf{b})$$

We will stick to this RELATIONAL STYLE (as opposed to the FUNCTIONAL STYLE) throughout the book as much as possible. Thus instead of:

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

we will represent the denotation of a transitive verb in lambda calculus as:

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

This expression 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 '$\lambda x.\lambda y.\,x$ loves y', with a blend of English and lambda calculus.

For consistency, we define upper-case $\lambda x.\mathsf{Happy}(x)$ to be equivalent to $\lambda x.\mathsf{happy}(x)$, where happy is a predicate of type $\langle e,t\rangle$, and so on for each predicate. So each lower-case predicate will have a matching upper-case predicate that we will make use of. The upper-case predicate and relation symbols are not part of the official language (so the interpretation function in models of this language will not have to worry about both variants), but we will use the upper-case predicates in practice when it is convenient.

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:

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

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

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

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 $\langle e, e\rangle$:

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

This function takes as input an individual x and returns as output another individual, the 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 $\langle\sigma,\tau\rangle$.

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 $[\![\lambda u.\alpha]\!]^{M,g}$ 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 1 (True) if o is happy, and 0 (False) if not. So $\lambda x.\mathsf{Happy}(x)$ denotes the characteristic function of the set of happy individuals. 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

$$[\lambda x.\mathsf{Loves}(x,\mathsf{bj})](\mathsf{ag})$$

denotes the result of applying the function 'loves Björn' to Agnetha. (We will often drop the square brackets when it does not result in confusion.)

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

$$\mathsf{Loves}(\mathsf{ag},\mathsf{bj})$$

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).

The following pairs of expressions are equivalent; where the second is the beta-reduced version of the first.

(1) a. $[\lambda x.\mathsf{Smiled}(x)](\mathsf{a})$
b. $\mathsf{Smiled}(\mathsf{a})$

(2) a. $[\lambda x.[\mathsf{Smiled}(x) \wedge \mathsf{Happy}(x)]](\mathsf{a})$
b. $[\mathsf{Smiled}(\mathsf{a}) \wedge \mathsf{Happy}(\mathsf{a})]$

(3) a. $[\lambda x.[\mathsf{Smiled}(x) \wedge \mathsf{Happy}(y)]](\mathsf{a})$
b. $[\mathsf{Smiled}(\mathsf{a}) \wedge \mathsf{Happy}(y)]$

In a lambda expression of the form $\lambda x.\phi$, the ϕ part (the scope of the lambda expression) describes the value of the function given an argument, so it can be called the VALUE DESCRIPTION (or BODY). For example, the value description in the expression

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

is

$$\mathsf{Loves}(x,\mathsf{bj})$$

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 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 occurrences of x that are free 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.

We say "all *free* occurrences" because if another variable binder is present in the value description and binds 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:

(4) 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 rule 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

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

into

(6) $\lambda z.\mathsf{Loves}(z, y)$

Here we replaced x with z, which we could do because z did not already occur free in the variable description in (5). 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:

(7) $[\lambda y.\lambda x.\mathsf{Loves}(x,y)](x)$

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

(8) $\lambda x.\mathsf{Loves}(x,x)$

But this is not equivalent to the original expression. It denotes the set of self-lovers, rather than the set of those who love x. 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:

(9) $[\lambda y.\lambda z.\mathsf{Loves}(z,y)](x)$

and this would have had exactly the same denotation as (7). If we perform beta reduction on (9), then we get the right result:

(10) $\lambda z.\mathsf{Loves}(z,x)$

Whereas our former attempt in (8) 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 (7), we recommend first using alpha conversion to 're-letter' the bound variable as in (9), before carrying out beta reduction as usual. This recommendation is enforced in the Lambda Calculator, as you will discover through practice.[2]

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:

$$\lambda x.\mathsf{Smokes}(x)$$

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

$$\lambda x.\neg\mathsf{Smokes}(x)$$

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:

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

[2] A third rule, ETA REDUCTION, allows us to rewrite a lambda term like $\lambda x.\mathsf{smiles}(x)$ simply as smiles. This can be another handy way of simplifying representations. Like the other rules, it comes with provisos regarding free variables. For example, $\lambda x.\mathsf{loves}(x)(x)$ (denoting the set of self-lovers) cannot be eta-reduced to $\mathsf{loves}(x)$, which denotes the set of x-lovers rather than self-lovers. So the lambda-bound variable cannot be free in the product of an eta-conversion.

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.

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:

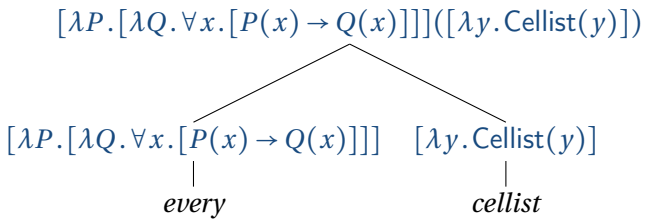
$$\lambda P.[\lambda Q.\forall x.[P(x)\rightarrow Q(x)]]$$

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 1 (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:

$$\lambda y.\mathsf{Cellist}(y)$$

Then *every cellist* will be translated as:



The translation at the top beta-reduces to:

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

which in turn beta-reduces to:

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

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:

$$[\lambda Q.\forall x.[\mathsf{Cellist}(x) \rightarrow Q(x)]]([\lambda z.\mathsf{Smokes}(z)])$$

$$\equiv \forall x.[\mathsf{Cellist}(x) \rightarrow \mathsf{Smokes}(x)]$$

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.

Under the hood, we will simply assume that there are an infinite set of constants and variables of every type.[3] Outside of the official syntax, we will allow ourselves to define particular constants like Happy and motherOf to be equivalent to one of the actual constants in the language.

1. **Basic Expressions**
   For every natural number n and every type τ, there is:
   - a constant of the form $c_{\tau,n}$
   - a variable of the form $v_{\tau,n}$

2. **Application**
   For any types σ and τ, if α is an expression of type $\langle \sigma, \tau \rangle$ and β is an expression of type σ then $[\alpha](\beta)$ is an expression of type τ. (We will often drop the square brackets when it does not result in confusion.)

3. **Equality**
   If α and β are of the same type, then $\alpha = \beta$ is an expression of type t.

[3] This choice is not crucial; we are following Dowty et al. (1981) in this respect, but another alternative would be to follow **?**, II, p. 119, who assume a possibly empty set of constants for each type. Thanks to Magdalena Kaufmann for discussion of this point.

4. **Negation**
   If ϕ is a formula, then so is $\neg\phi$.

5. **Binary Connectives**
   If ϕ and ψ are formulas, then so are $\neg\phi$, $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \to \psi]$, and $[\phi \leftrightarrow \psi]$.

6. **Quantification**
   If ϕ is a formula and u is a variable of any type, then $[\forall u\,.\,\phi]$ and $[\exists u\,.\,\phi]$ are formulas.

7. **Lambda abstraction** (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. $\lambda x \lambda y\,.\,\mathsf{Admires}(x,y)$. 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, so we can for example write $\pi(\lambda x\,.\,x+1)$ rather than $\pi([\lambda x\,.\,\mathsf{Happy}(x)])$: Square brackets that are immediately embedded inside parentheses can be dropped.

Finally, we define some equivalences between 'relational style' and 'functional style' formulas. For example,

$$\mathsf{Loves}(x,y)$$

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

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

can be re-written as

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

where Π is a variant of π that starts with a capital letter.

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

22. $[\lambda P \lambda x.\neg P(x)](\lambda x[\mathsf{mortal}(x)])$

23. $[\lambda P \lambda x.P(x)](\lambda x[\neg \mathsf{mortal}(x)])$

You can check your answers using the Lambda Calculator.

5.3.2 Semantics

As in L_1, the semantic values of expressions in L_λ depend on a model and an assignment function. As in L_1, a model $M = \langle D, I \rangle$ is a pair consisting of the domain of individuals D and an interpretation function I, which assigns semantic values to each of the non-logical constants in the language. For every type τ, I assigns an object of type τ to every non-logical constant of type τ.

Recall that types are associated with domains:

- The domain of individuals D_e is the set of individuals, the set of potential denotations for an expression of type e.
- The domain of truth values D_t contains just two elements: 1 'true' and 0 'false'.
- For any types σ and τ, $D_{\langle \sigma,\tau \rangle}$ is the domain of functions from D_σ to D_τ.

Assignments provide values for variables of all types, not just those of type e. An assignment thus is a function assigning to each variable of type τ a denotation from the set D_τ.

The semantic value of an expression is defined as follows:

1. **Basic Expressions**
   (a) If α is a non-logical constant, then $[\![\alpha]\!]^{M,g} = I(\alpha)$.
   (b) If α is a variable, then $[\![\alpha]\!]^{M,g} = g(\alpha)$.

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

3. **Equality**
   If α and β are expressions of the same type, then $[\![\alpha = \beta]\!]^{M,g} = 1$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$.

4. **Negation**
   If ϕ is a formula, then $[\![\neg\phi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 0$.

5. **Binary Connectives**
   If ϕ and ψ are formulas, then:

   (a) $[\![\phi \wedge \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$.
   (b) $[\![\phi \vee \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 1$ or $[\![\psi]\!]^{M,g} = 1$.
   (c) $[\![\phi \rightarrow \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 0$ and $[\![\psi]\!]^{M,g} = 1$.
   (d) $[\![\phi \leftrightarrow \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = [\![\psi]\!]^{M,g}$.

6. **Quantification**

   (a) If ϕ is a formula and v is a variable of type τ then $[\![\forall v\,.\,\phi]\!]^{M,g} = 1$ iff for all objects $o \in D_\tau$:

   $$[\![\phi]\!]^{M,g[v\mapsto o]} = 1$$

   (b) If ϕ is a formula and v is a variable of type τ then $[\![\exists v\,.\,\phi]\!]^{M,g} = 1$ iff there is some object $o \in D_\tau$ such that:

   $$[\![\phi]\!]^{M,g[v\mapsto o]} = 1$$

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 now!* 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. The answer should 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

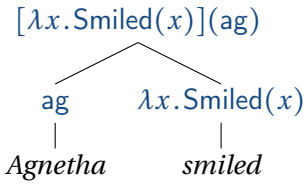
6.1 Introduction

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 the 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.

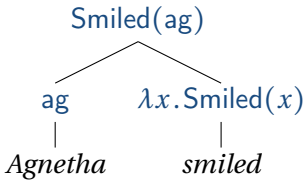
How, then, do denotations of constituents compose? We will first explore the hypothesis, inspired by Frege's idea of saturation, that all composition is function application. In this chapter, we will define a semantics for a fragment of English that adheres to this principle. To do so, we will *translate* expressions of English into expressions of L_λ. A name like *Agnetha* will translate as the type e expression ag; both denote the individual Agnetha. The intransitive verb will be translated as the type $\langle e, t\rangle$ expression $\lambda x.\mathsf{Smiled}(x)$; both denote the set of smilers.

- *Agnetha* $\rightsquigarrow \mathsf{ag}$
- *smiled* $\rightsquigarrow \lambda x.\mathsf{Smiled}(x)$

Here, $\rightsquigarrow$ signifies the 'translates to' relation. The combination, *Agnetha smiled*, will then be translated as the result of applying the translation of the verb to the translation of the subject:



... or equivalently, through beta reduction:



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:

$$[\![\textit{Agnetha smiled}]\!] = 1$$

we instead write:

$$\textit{Agnetha smiled} \rightsquigarrow \mathsf{Smiled}(\mathsf{ag})$$

and:

$$[\![\mathsf{Smiled}(\mathsf{ag})]\!] = 1$$

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).

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 (1973a) 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, 1970), 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 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

[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.

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 (1970) formally defined the first fragment of English, consisting of:

- 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 lexical entries;
- a list of composition rules.

Throughout this book we too will build up a fragment in a similar style.

[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:

$$[\![\textit{snores}]\!] = \lambda x\,.\,x \text{ snores}$$

with a mix of English and lambdas on the right-hand side of the equation. In contrast, we would write equations mapping object language to representation language like this:

$$\textit{snores} \rightsquigarrow \lambda x\,.\,\mathsf{snores}(x)$$

and equations mapping representation language to denotations specified in the meta-language like this:

$$[\![\lambda x\,.\,\mathsf{snores}(x)]\!]^{M,g} = I(\mathsf{snores})$$

One should carefully distinguish between these two ways of using the λ symbol and make sure to be consistent.

We already have our representation language: L_λ as defined in the previous chapter. The next step is to specifying 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.[3]

(1) **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 con-

[3]Here we are following the convention known as the 'DP-hypothesis', where DP stands for 'Determiner Phrase', and we assume that the head of a phrase like *the car* is the determiner *the*. Since phrases like *the car* and proper names like *Agnetha* are of the same category, this means that *Agnetha* is also a DP. Rather than treating proper names directly as DPs, we will treat them as intransitive D's, since D is a lexical category. Just as there are transitive and intransitive verbs, there are also transitive and intransitive variants of other categories. For example, particles like *up* as in *eat it up* can be viewed as intransitive prepositions. Here we are assuming that D's can be intransitive as well. Longobardi (1994) assumes that proper names start out as category N and move to D; here we are just analyzing them directly as D's. Their semantics will prevent them from combining with NPs the way that other D's do.

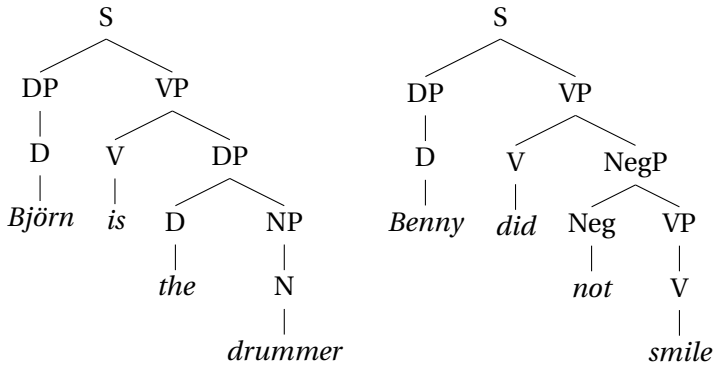
sist solely of a verb, or of a verb followed by an NP, or of a verb followed by an AP, etc.

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:

(2) **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:



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.

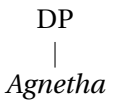
(i) Frida smiles with nobody.

(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:



instead of

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, we have only one rule for composing the denotation of a complex expression out of the denotations of the parts, namely FUNCTION APPLICATION, which 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 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, where:

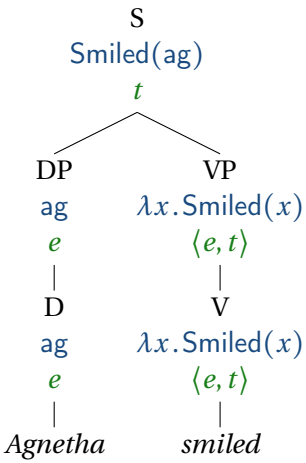
- $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where β' has type σ.

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):



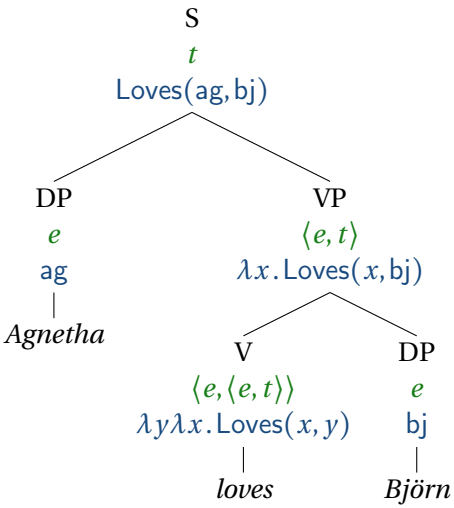
6.2 Fun with Function Application

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:

(3) *loves* $\rightsquigarrow \lambda y \lambda x . \mathsf{Loves}(x, y)$

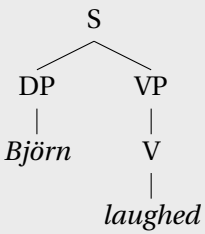
Can this verb combine semantically with a type-e direct object via Function Application? Yes, it can; the types match. This is shown in the following derivation for *Agnetha loves Björn*:



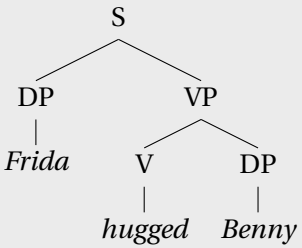
Via Function Application, the transitive verb *loves* combines with the object *Björn*. The VP *loves Björn* thus comes to denote (the characteristic function of) the set of all individuals who love Björn, which we can think of as the property of loving Björn. This property is then attributed to Agnetha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

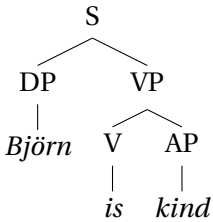


(b)



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:



We will continue to assume that the proper name *Björn* is translated as the constant bj, 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.

(4) $kind \rightsquigarrow \lambda x.\mathsf{Kind}(x)$

Now, what is the contribution of *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 starting dealing with tense yet, we will ignore the former function and focus on the latter. We can capture the fact that *is* connects

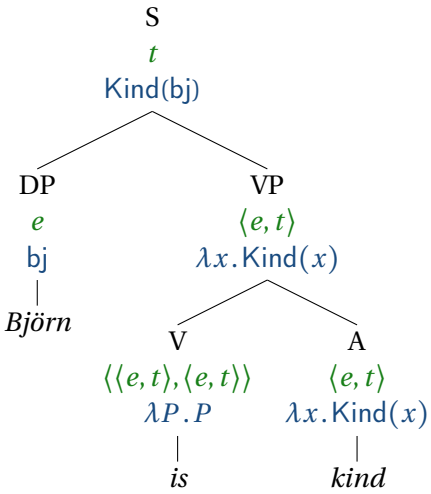
the predicate to the subject by treating it as an IDENTITY FUNCTION, a function that returns whatever it takes in as input. In this case, *is* takes in a function of type $\langle e, t\rangle$, and returns that same function. (We adopt the same approach for other words that seem to lack meaning of their own, such as *did* in *Benny did not smile*.)

(5) $\textit{is} \rightsquigarrow \lambda P\,.\,P$

This implies that *is* denotes a function that takes as its first argument another function P, where P is of type $\langle e, t\rangle$, and returns P.

With these rules, we will end up with the following analysis for the sentence *Björn is kind*:

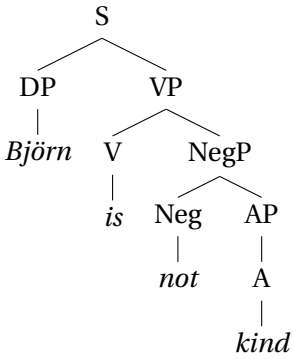
(6)



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 A). The individual lexical entries that we have specified are used at the terminal nodes (*Björn*, *is*, and *kind*).

6.2.3 *Björn is not kind*

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:



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

$$\neg\mathsf{Kind}(\mathsf{bj})$$

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

$$\lambda x.\neg\mathsf{Kind}(x)$$

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:

(7) $not \rightsquigarrow \lambda P\lambda x.\neg P(x)$ transl 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. Note that in this lambda expression, the value description is $\lambda x.\neg P(x)$, so the return value is itself another function.

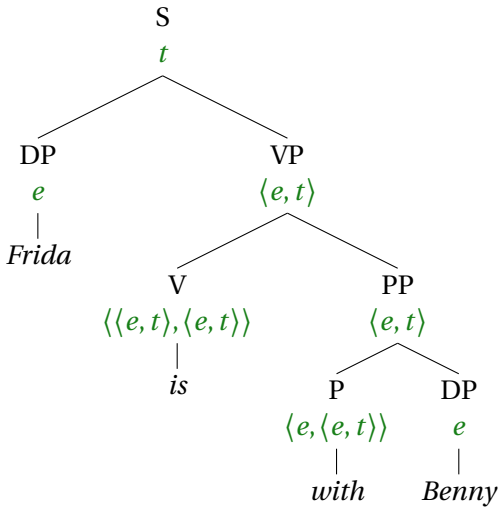
Exercise 4. Using this lexical entry for *not*, give a compositional analysis of *Björn is not kind*, by showing the translations and types at each node of the syntax tree.

6.2.4 *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:

(8) $\textit{with} \rightsquigarrow \lambda y \lambda x . \mathsf{With}(x, y)$

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.



Exercise 5. Derive the translation into L_λ for *Frida is with Benny* by giving a fully beta-reduced translation for each node.

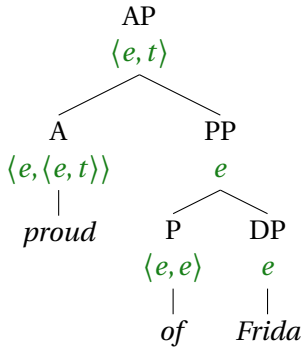
6.2.5 *Benny is proud of Frida*

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 $\langle e, e\rangle$.

(9) $\textit{of} \rightsquigarrow \lambda x . x$

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



Exercise 6. 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.)

6.2.6 *Agnetha is a singer*

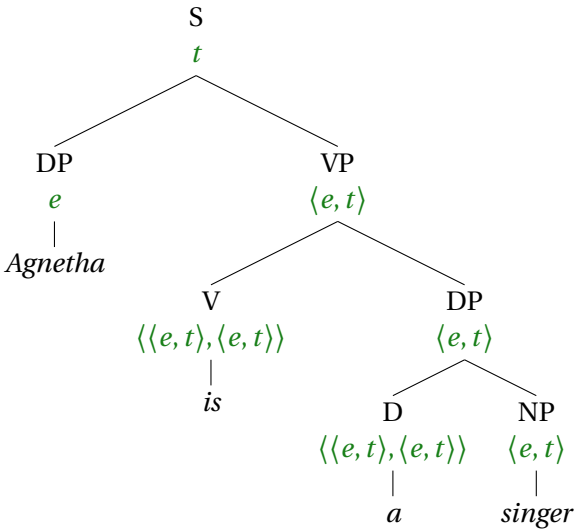
Let us consider *Agnetha is a singer*. The noun *singer* can be analyzed as an $\langle e, t\rangle$ type property like *Swedish*, the characteristic function of the set of individuals who are singers.

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.

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

With these assumptions, the derivation will go as follows.

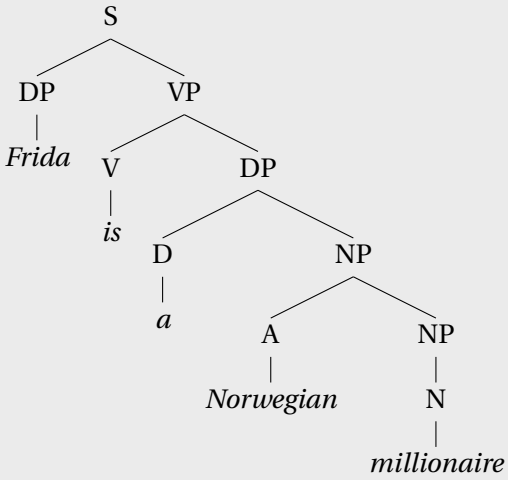
(11)



Exercise 7. Give fully beta-reduced translations at each node of the tree for *Agnetha is a singer*.

Exercise 8. 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 9. 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?



6.3 Quantifiers: Type $\langle\langle e,t\rangle,t\rangle$

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

(12) Everybody smiled.

We have assumed that a VP like *smiled* denotes a predicate (type $\langle e, t\rangle$) and that a sentence like (12) denotes a truth value (type t). Based on what we established in the chapter on quantification in predicate logic, the translation of *Everybody smiled* should be something like the following (assuming that every individual in the domain is conceived of as human):

(13) *everybody smiled* $\rightsquigarrow \forall x.\mathsf{Smiled}(x)$

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

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

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:

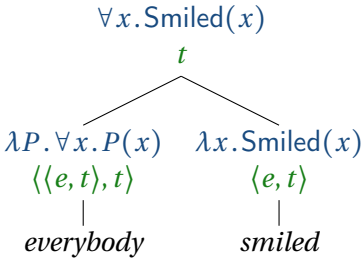
(14) *everybody* $\rightsquigarrow \lambda P.\forall x.P(x)$

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:

$$\langle\langle e, t\rangle, t\rangle$$

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:



In this derivation, the *VP* is fed as an *argument* to the *subject DP*. 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.

We can define *something* and *nothing* similarly:

(15) $\textit{something} \rightsquigarrow \lambda P.\exists x.P(x)$

(16) $\textit{nothing} \rightsquigarrow \lambda P.\neg\exists x.P(x)$

All of these quantifiers can be thought of as predicates of predicates. For example, $\lambda P.\neg\exists x.P(x)$ denotes a predicate that holds of a predicate over individuals if it has no satisfiers. We turn now to a more detailed motivation for this view.

6.3.1 Quantifiers: not type *e*

Previously, we analyzed indefinite descriptions like *a singer* in sentences like *Agnetha is a singer*. In this type of example, the indefinite description functions as a PREDICATE, applies to a subject. An indefinite description can also function as the subject or object of a transitive verb, as in the following sentences:

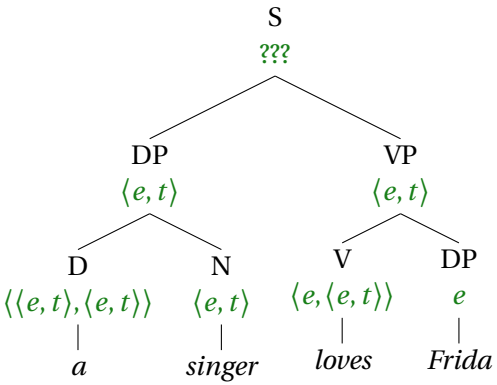
(17) a. A singer loves Frida. [subject position]
b. Frida loves a singer. [object position]

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 particular type of syntactic position within a sentence.)

So far, we do not have the tools to analyze indefinite descriptions in argument position. We have analyzed the denotation of *a* as an identity function on predicates, so *a singer* denotes just the same thing as *singer*. Both have (a translation with) type $\langle e, t\rangle$. In a sentence like *A singer loves Frida*, the VP *loves Frida* translates to an expression of type $\langle e, t\rangle$.

This leaves us in the following predicament.



We have no composition rules for combining two expressions of type $\langle e, t\rangle$.

We also lack an analysis for the following sentences.

(18) Somebody is brave.

(19) Everybody is brave.

(20) Nobody is brave.

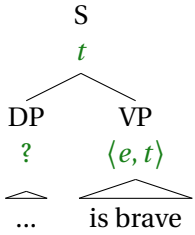
(21) {Some, every, at least one, at most one, no} linguist is brave.

(22) {Few, some, several, many, most, more than two} linguists are brave.

We have been assuming that the VP has a translation of type $\langle e, t\rangle$. A sentence can be true or false, so it should have a translation of

type t. Given this, what type could the subject get in these examples?

(23)



We could arrive at type t at the S node by giving these expressions a translation of type e, like *Frida*, *Agnetha*, and *the baby*. But the expressions in (21) and (22) expressions cannot be of type e. We can *show* that they cannot be of type e based on certain empirical facts.

Any expression of type e should have certain properties that the expressions in question lack. An expression of type e denotes a particular individual, so two occurrences of the same expression of type e denote the same individual. (Pronouns like *him* and *her*, 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. Here we are setting aside expressions like these.)

An expression of type e should validate **subset-to-superset inferences**, for example:

(24) Susan came yesterday morning.
∴ Susan came yesterday.

This is a valid inference if the subject, like *Susan*, denotes an individual. Here is why. The set of things that came yesterday morning is a subset of the things that came yesterday. 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.

Some of the expressions in (18)–(22) fail to validate subset-to-superset inferences. For example:

(25) At most one letter came yesterday morning.
∴ At most one letter came yesterday.

This inference is *not* valid, so *at most one letter* cannot denote an individual, so it cannot be of type *e*.

Exercise 10. Which of the expressions in (21) validate subset-to-superset inferences? Give examples.

Another property that expressions of type *e* should have that these expressions do not always have is adherence to the LAW OF NON-CONTRADICTION. In logic, the law of non-contradiction is the principle that $[p \wedge \neg p]$ is false for any p. If we take the sentence *Mont Blanc is higher than 4,000m* and conjoin it with its negation, the result is self-contradictory:

(26) 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. Incidentally, the same reasoning works in reverse; if the second conjunct is true, then the first must be false. The conjunction (under an analysis of *and* as logical conjunction) can therefore never be true.

The following sentence, however, is not contradictory:

(27) More than two mountains are higher than 4,000m, and more than two mountains are not higher than 4,000m.

Hence, *more than two mountains* cannot have type e.

Exercise 11. Which of the expressions in (21) fail to validate the law of non-contradiction? Give examples.

Exercise 12. 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: ...")

Upon inspection, you will find that each of the subject noun phrases in (18)–(22) can be shown not to be of type e. Evidently, they must be of some other type.

6.3.2 Solution: Predicates of predicates

Let us recap. We assume that a VP denotes a predicate (type $\langle e, t\rangle$) and that a sentence denotes a truth value (type t). We have a bunch of expressions that are not of type e, and we want them to combine with the VP to produce something of type t. What can we do? The solution (due to Montague) is to feed the *VP* as an *argument* to the *subject DP* instead of the other way around. A DP like *something* will denote a function that takes a predicate as an argument, and returns a truth value. Its type will therefore be:

$$\langle\langle e, t\rangle, t\rangle$$

This is the type of a QUANTIFIER.

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, *something* can be seen as denoting a function that takes as input a predicate and returns true iff there is at least one thing that satisfies the predicate:

(28) $\textit{something} \rightsquigarrow \lambda P . \exists x . P(x)$

In contrast, the function denoted by *nothing* returns true iff there is nothing satisfying the predicate:

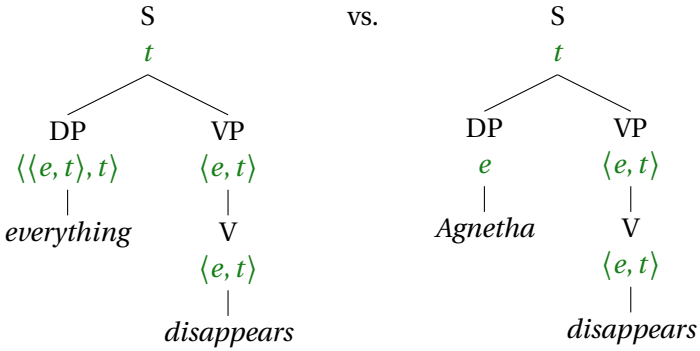
(29) $\textit{nothing} \rightsquigarrow \lambda P . \neg \exists x . P(x)$

The function denoted by *everything* returns true iff everything satisfies the predicate:

(30) $\textit{everything} \rightsquigarrow \lambda P . \forall x . P(x)$

Using Function Application (recall that it does not care about the order of the arguments), the quantifier will take the denotation of the VP as an argument, and yield a truth value, thus:

(31)



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:

$$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$$

These determiners can be defined in terms of the quantifiers $\exists$ and $\forall$ of first-order logic:

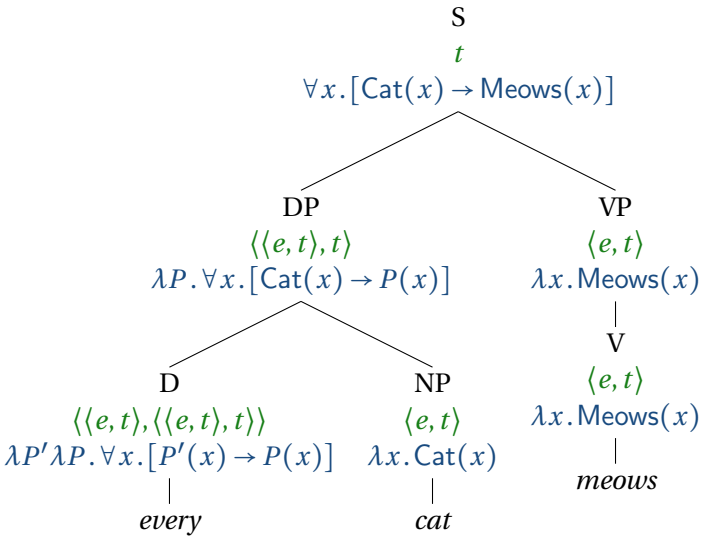
(32) *some* $\rightsquigarrow \lambda P'\lambda P.\exists x.[P(x)\wedge P'(x)]$

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

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

These lexical entries will yield analyses like the following:

(35)



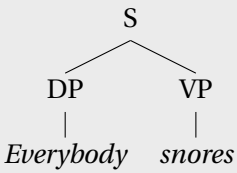
Exercise 13. Give an analysis of *A singer loves Frida* using Functional Application and Non-Branching Nodes. Your analysis should take the form of a tree, specifying at each node, the syntactic category, the semantic type, and a fully beta-reduced translation to L_λ. The translation of the sentence 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*.

Exercise 14. 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:

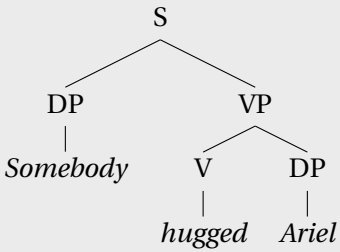
- Adjectives, non-relational common nouns, and intransitive verbs are of type $\langle e, t\rangle$.
- Transitive verbs are of type $\langle e, \langle e, t\rangle\rangle$.
- Proper names are of type e.
- Quantificational DPs are of type $\langle\langle e, t\rangle, t\rangle$.
- Determiners are of type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

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.

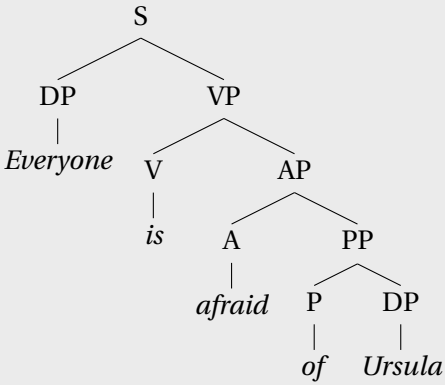
(a)



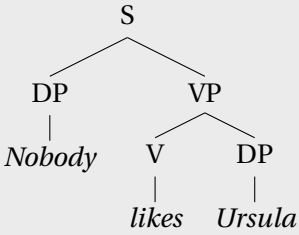
(b)


S
DP
VP
Somebody
V
DP
hugged
Ariel

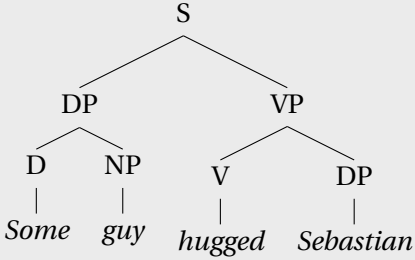
(c)


S
DP
VP
Everyone
V
AP
is
A
PP
afraid
P
DP
of
Ursula

(d)


S
DP
VP
Nobody
V
DP
likes
Ursula

(e)


S
DP
VP
D
NP
V
DP
Some
guy
hugged
Sebastian

Exercise 15. How does the kind of treatment of quantificational expressions given in the preceding discussion account for these facts:

(a) *More than two cats are indoors and more than two cats are not indoors* is not a contradiction.

(b) *Everybody here is over 30 or everybody here is not over 30* is not a tautology.

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 Generalized Quantifiers

(This section is under development.)

We have said that *every cat* translates to the following expression of type $\langle\langle e, t\rangle, t\rangle$:

(36) *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, (36) 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: (36) 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:

(37) $\{P \subseteq D_e : \mathsf{CAT} \subseteq P\}$

Similarly, *some dog* translates to this:

(38) *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:

(39) $\{P \subseteq D_e : P \cap \mathsf{DOG} \neq \varnothing\}$

We will call these sets $\mathsf{everyCat}$ and $\mathsf{someDog}$. They can be visualized as in Figures 6.1 and 6.2.

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

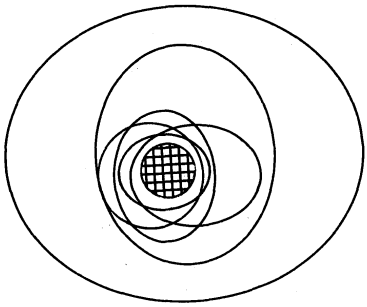


Figure 6.1: The generalized quantifier everyCat denoted by *every cat*. The biggest circle represents the universe of discourse. The cross-hatched circle represents the set of cats. The other circles represent some of the sets in the denotation of the generalized quantifier. The cross-hatched circle must be fully contained in each of the other circles because they represent properties of *every* cat, that is, supersets of the set of cats.

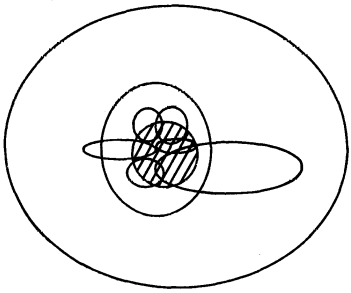


Figure 6.2: The generalized quantifier someDog denoted by *some dog*. The biggest circle represents the universe of discourse. The dashed circle represents the set of dogs. The other circles represent some of the sets in the denotation of the generalized quantifier. The cross-hatched circle need not 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:

(40) a. $\text{everyThing} =_{def} \{P \subseteq D_e : D_e \subseteq P\} = \{D_e\}$
b. $\text{someThing} =_{def} \{P \subseteq D_e : P \cap D_e \neq \varnothing\}$
$= \{P \subseteq D_e : P \neq \varnothing\}$

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*.

(41) a. $\text{noThing} =_{def} \{P \subseteq D_e : |P| = 0\} = \{\varnothing\}$
b. $\text{exactlyTwoThings} =_{def} \{P \subseteq D_e : |P| = 2\}$
c. $\text{atLeastTwoThings} =_{def} \{P \subseteq D_e : |P| \geq 2\}$

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 (41c). Otherwise, it denotes exactlyTwoThings in (41b).

Sets of sets of entities like those in (41) 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 (36) and *some dog* in (38), as well as the quantifiers in (41). Some of these look very simple:

(42) *nothing* $\rightsquigarrow \lambda P.\neg\exists x.P(x)$

Others look quite unwieldy:

(43) *exactly two things* $\rightsquigarrow \exists x.\exists y.\ \neg(x=y)\wedge P(x)\wedge P(y)\wedge \neg\exists z.P(z)\wedge\neg(z=x)\wedge\neg(z=y)$

Other generalized quantifiers, such as those denoted by *most swans*, *most things*, *one in three cats* are not first-order definable. But they can be defined in terms of sets:

(44) 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. $\mathsf{oneInThreeCats} =_{def} \{P\subseteq D_e : |P\cup\mathsf{CAT}|/|\mathsf{CAT}|\geq 1/3\}$

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 (41) 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 has a compositional structure that corresponds roughly to *more* + *-st*.) 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*:

(45) $\textit{every} \rightsquigarrow \lambda P'\lambda P.\forall x.P'(x) \rightarrow P(x)$

The Det that this denotes is the left-to-right Curried version of the subset relation. We will refer to this relation as every:

(46) $\mathsf{every} =_{def} \{\langle P', P\rangle \mid P' \subseteq P\}$

Likewise, the translation of *some* in (47) denotes the Curried version of the relation in (48), which holds between any two sets when they overlap:

(47) $\textit{some} \rightsquigarrow \lambda P'\lambda P.\exists x.P'(x) \wedge P(x)$

(48) $\mathsf{some} =_{def} \{\langle P', P\rangle \mid P' \cap P \neq \varnothing\}$

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*.[4]

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 $\forall$:

(49) a. $\mathsf{some}(\lambda x.\mathsf{Dog}(x))(\lambda x.\mathsf{Barks}(x))$
b. $\exists x.\mathsf{Dog}(x) \wedge \mathsf{Barks}(x)$

(50) 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*:

(51) a. Some A is a B $\Leftrightarrow$
b. Some entity is both an A and a B.

(52) a. Every A is a B $\Leftrightarrow$
b. Every entity is such that, if it is an A, then it is a B.

[4] 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).

A Det that is not sortally reducible is called INHERENTLY SORTAL. An example is most, which cannot be paraphrased in the required way:

(53) 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 (53a) is intuitively judged false. But sentence (53b), 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 6.3 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.

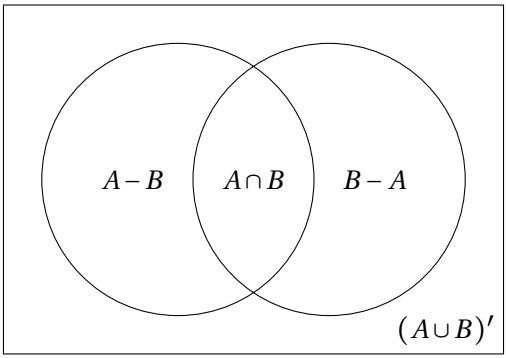


Figure 6.3: The four sets that a Det can depend on.

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 *A*s without being *B*s. 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 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 (46), every is intended to mirror the behavior of $\forall$ in first-order logic, and therefore lacks existential import.

Exercise 17. Define a version of every that has existential import and call it $\text{every}_{\exists}$.

A CO-INTERSECTIVE Det depends on $A - B$ and nothing else. So every as defined in (46) is CO-INTERSECTIVE, while $\text{every}_{\exists}$ as defined in Exercise 17 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. 6.3 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 (54), which violates conservativity but satisfies extension:

(54) $\{\langle A, B\rangle \mid |A| = |B|\}$

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:

(55) _____ 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:

(56) 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*:

(57) 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:

(58) $\{\langle A, B\rangle \mid B \subseteq A\}$

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:

(59) 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:

(60) 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.

(61) 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*:

(62) 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

(63) 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 DEFINITE, 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.

Exercises

Exercise 18. 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 6.4. 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.

Exercise 19. 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 tran-

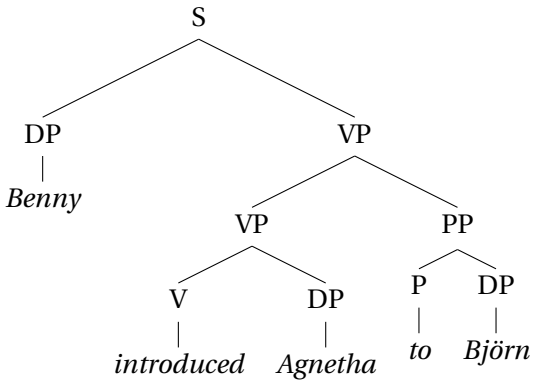
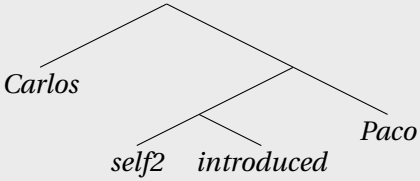


Figure 6.4: A ditransitive verb

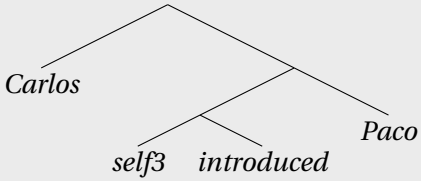
sitive verb, translated as an expression of type $\langle e, \langle e, t \rangle \rangle$. You can use the lexical entry for *introduce* given in the previous exercise.

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

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



(c) For the sentence *Carlos self3-introduced Paco*, make the structure below yield the denotation 'Carlos introduced Paco to Paco (himself)' by supplying the denotation of *self3*.



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

(Exercise due to Maribel Romero.)

6.4.1 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:
  - $\alpha \rightsquigarrow \alpha'$ where α' has type $\langle \sigma, \tau \rangle$
  - $\beta \rightsquigarrow \beta'$ where β' has type σ.

  Then
  $$\gamma \rightsquigarrow \alpha'(\beta')$$

- **Non-branching Nodes** (NN)
  If β is a tree whose only daughter is α, where $\alpha \rightsquigarrow \alpha'$, then $\beta \rightsquigarrow \alpha'$.

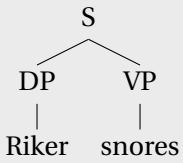
Lexical entries

- *Agnetha* $\rightsquigarrow \mathsf{ag}$
- *smiled* $\rightsquigarrow \lambda x.\mathsf{Smiled}(x)$
- *loves* $\rightsquigarrow \lambda y \lambda x.\mathsf{Loves}(x, y)$
- *kind* $\rightsquigarrow \lambda x.\mathsf{Kind}(x)$
- *is* $\rightsquigarrow \lambda P.P$

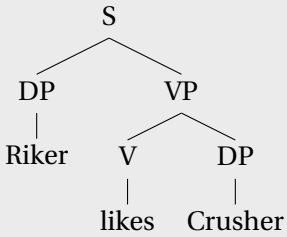
- *with* $\rightsquigarrow \lambda y \lambda x . \mathsf{With}(x, y)$
- *of* $\rightsquigarrow \lambda x . x$
- *a* $\rightsquigarrow \lambda P . P$
- *not* $\rightsquigarrow \lambda P \lambda x . \neg P(x)$

Exercise 20. Give fully beta-reduced translations at each node of the following trees. Provide appropriate lexical entries as needed.

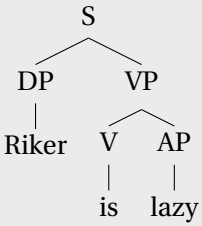
(a)



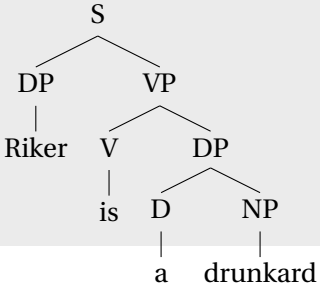
(b)



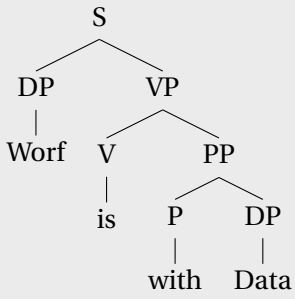
(c)



(d)

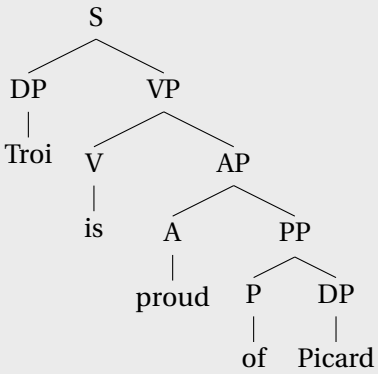


(e)

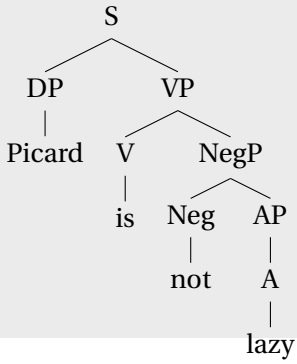


Exercise 21. Give fully beta-reduced translations at each node of the following trees. Provide appropriate lexical entries as needed.

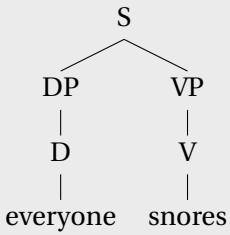
(a)



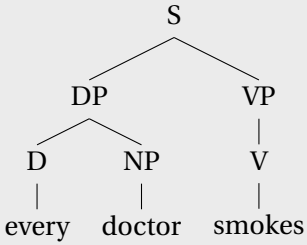
(b)



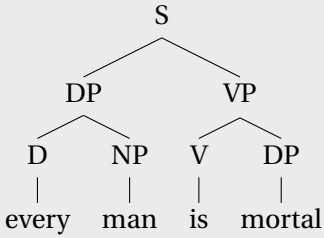
(c)



(d)



(e)



Exercise 22. 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 23. 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.

- *Björn smokes and Agnetha 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)
∴ This doubt is reasonable. (Conclusion, predicative use)

(4) This doubt is reasonable. (Premise, predicative use)
∴ This is a reasonable doubt. (Conclusion, attributive use)

Dropping this adjective when it occurs in attributive position also results in a valid argument:

(5) This is a reasonable doubt. (Premise)
∴ This is a doubt. (Conclusion)

And we can reason from the adjective and the noun to their combination:

(6) This is reasonable. (Premise 1)
This is a doubt. (Premise 2)
∴ 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 entailments:

(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 *reasonable* denotes a set—the set of all reasonable things—and that *doubt,* just like other nouns we have seen, de-

notes a set as well—the set of doubts. A reasonable doubt is something which is in each of these two sets, or in other words, in their intersection. For this reason, the adjective *reasonable*, and others like it, 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.
∴ Einstein is an outstanding violinist. (not valid!)

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 *outstanding violinist* denotes a subset of the denotation of *violinist*. In this sense, both *outstanding* and *physicist* are SUBSECTIVE ADJEC-

TIVES. 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.
∴ 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.
∴ 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. We will start with intersective adjectives, and come back to subsective and other adjectives.

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.)

(18) This is a reasonable doubt.
Reasonable(this) $\wedge$ Doubt(this)

(19) This is reasonable.
Reasonable(this)

(20) This is a doubt.
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 OutstandingAs of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$. For any type $\langle e,t\rangle$ expression P, OutstandingAs(P) is a new expression of type $\langle e,t\rangle$.

In order to ensure that 'x is an outstanding P' entails 'x is a P', we can stipulate that the following formula must be true every model:

(21) $\forall P \forall x.\,\text{OutstandingAs}(P)(x) \rightarrow P(x)$
(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 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 Bachelor is Male. 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.
$\mathsf{OutstandingAs}(P)(\mathsf{e})$

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 *simpliciter*, at least not in every context.

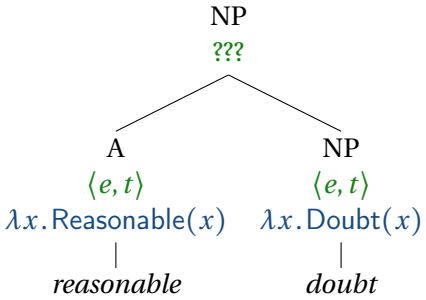
Exercise 1. Explain how the treatment of *outstanding* given above blocks the inference from *Einstein is an outstanding physicist* to *Einstein is an outstanding violinist.*

Having translated our sentences into logic, we have accounted for the entailment relations between them. But how do we make this translation compositional? 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 (18). In this sentence, the two expressions *reasonable* and *doubt* are sisters in the tree, but neither one denotes a function that has the de-

notation 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)

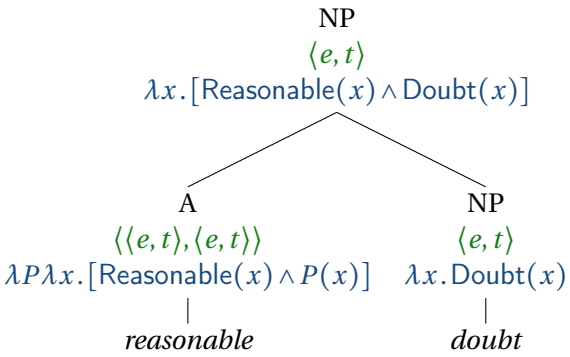


At this point, one might try and adopt a different denotation for *reasonable*, one that can be applied directly to *doubt*. This denotation would expect a predicate like *doubt*, and return a new predicate that holds of individuals that are both reasonable and in the set denoted by the input predicate. 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)



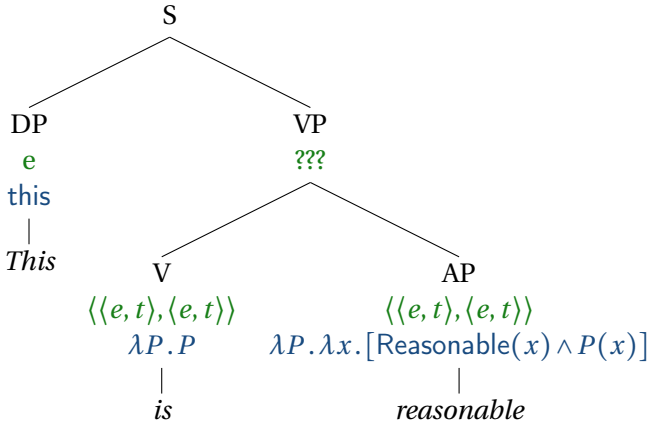
We will call this second translation for *reasonable* a MODIFIER and its type a MODIFIER TYPE.

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.

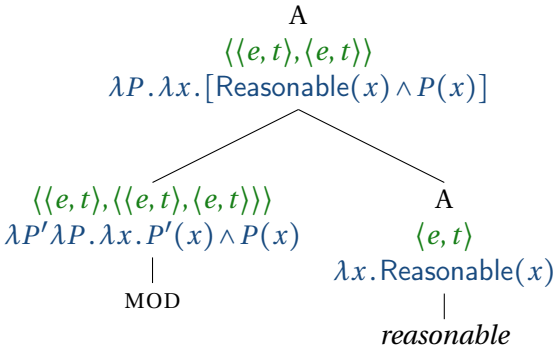
There are at least two ways to resolve this problem. We can (i) generate two translations for the adjective *reasonable* (and similarly for other intersective adjectives): one of type $\langle e,t\rangle$ for predicative positions, and another one of type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ 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 op-

erator MOD to generate derived translations, we will represent this as follows:

(30)

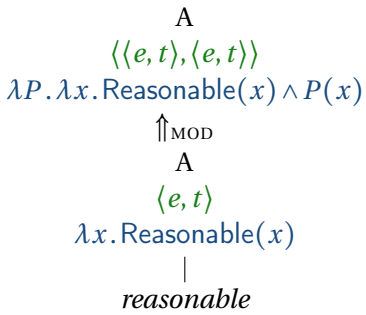


where MOD is an unpronounced word. Alternatively, we can use the following type-shifting rule to equivalent effect:

> **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)



Our notation uses the upwards-facing double arrow ⇑ 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 3. 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 $\langle e,t\rangle$

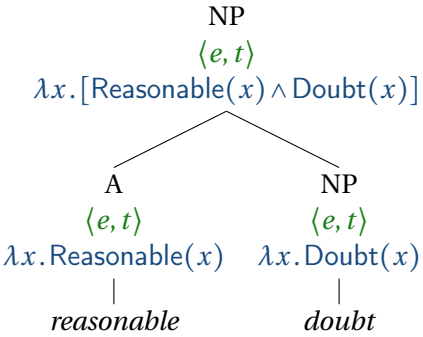
Then:

$$\gamma \rightsquigarrow \lambda u.[\alpha'(u) \wedge \beta'(u)]$$

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

(b) $\lambda x.x$

(c) $\lambda x.\mathsf{City}(x)$

(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).

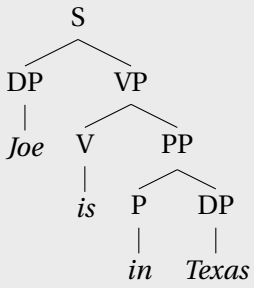
Exercise 5. In addition to the ones given above, adopt the following lexical entries, using the same assumptions about types as in the previous exercise:

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

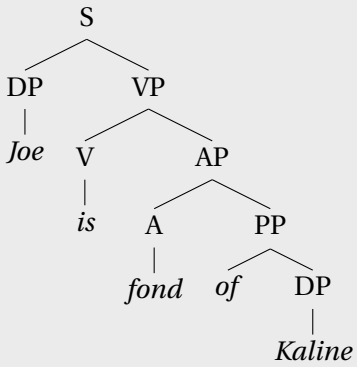
6. $in_2 \rightsquigarrow \lambda y \lambda P \lambda x . P(x) \wedge \mathsf{In}(x, y)$
7. $fond \rightsquigarrow \lambda y \lambda x . \mathsf{FondOf}(x, y)$
8. $fond_2 \rightsquigarrow \lambda y \lambda P \lambda x . P(x) \wedge \mathsf{FondOf}(x, y)$
9. *Joe* $\rightsquigarrow$ joe
10. *Texas* $\rightsquigarrow$ 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.

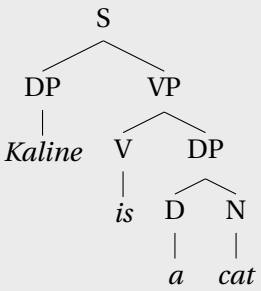
(a)



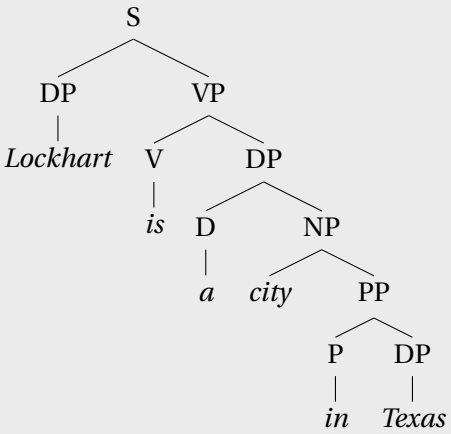
(b)



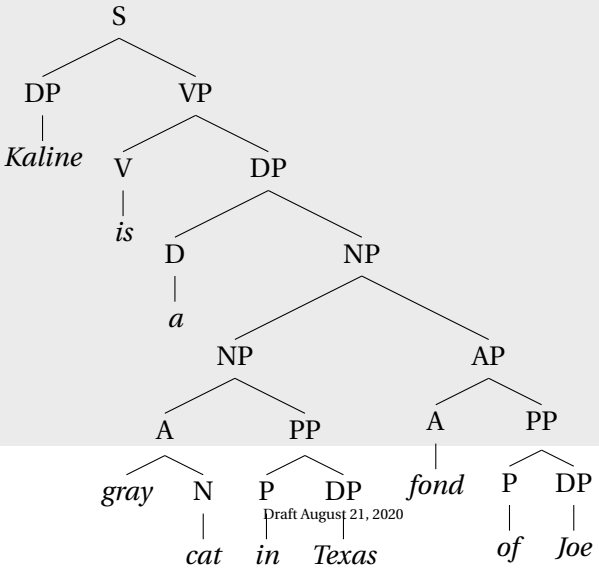
(c)


S
DP
VP
Kaline
V
DP
is
D
N
a
cat

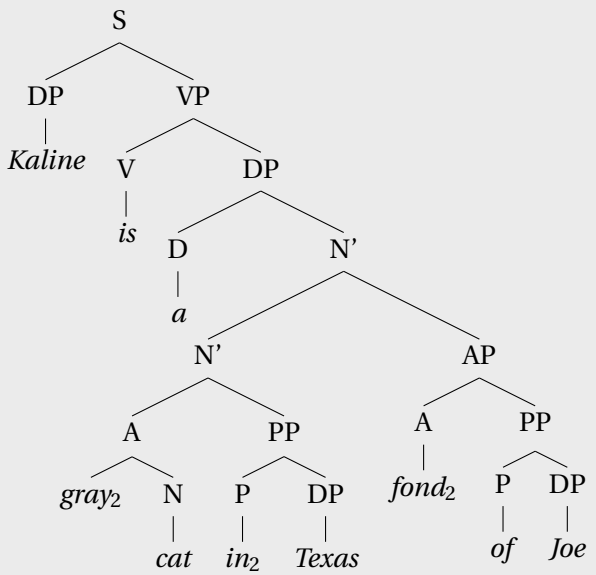
(d)


S
DP
VP
Lockhart
V
DP
is
D
NP
a
city
PP
P
DP
in
Texas

(e)


S
DP
VP
Kaline
V
DP
is
D
NP
a
NP
AP
A
PP
A
PP
gray
N
P
DP
fond
P
DP
cat
in
Texas
of
Joe

(f)



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) *yellow*

(b) *presumed*

(c) *future*

(d) *intelligent*

(e) *good*

(f) *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* in the feminine and *xoroš* in the masculine. The long form of 'good' is *xorošaja* in the feminine and *xorošij* in the 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) Èto byla xorošaja teorija.
this was good-LONG theory
'This was a good theory.'

(34) *Èto byla xoroša teorija.
this was good-SHORT theory
'This was a good theory.' (Intended.)

But in predicative positions, both forms are possible:

(35) Èta teorija byla xorošaja.
this theory was good-LONG
'This theory was good.'

(36) Èta teorija byla xoroša.
this theory was good-SHORT
'This theory was good.'

(37) Naša molodež' talantlivaja i trudoljubivaja.
our youth talented-LONG and industrious-LONG
'Our youth is talented and industrious.'

(38) Naša molodež' talantliva i trudoljubiva.
our youth talented-SHORT and 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 (39a), which involves an attributive use of the adjective *reasonable*, in terms of (39b), which involves a predicative use of the same adjective:

(39) 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:

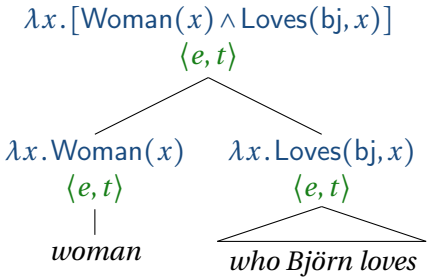
(40) woman who Björn loves

This expression characterizes an individual with 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 $\langle e, t\rangle$:

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

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:

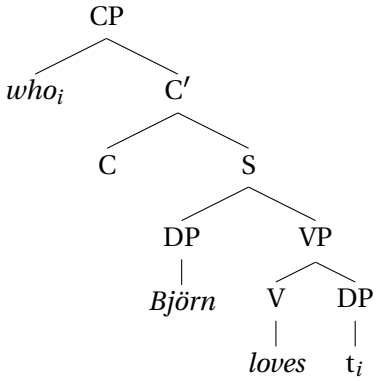


This expression captures the fact that a *woman who Björn loves* is both a woman and an individual loved by Björn.

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]

[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. 5).

[3]One reason to think that the word *that* is not of the same category as relative

(41) I woass ned **wann dass** da Xavea kummt.
I know not **when that** the Xavea comes
'I don't know when Xavier 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. To explain the fact that *that* and *which* cannot co-occur in English, we assume that either the relative pronoun or the complementizer *that* is deleted, in accordance with the 'Doubly-Filled Comp Filter' (Chomsky & Lasnik, 1977), the principle that either the relative pronoun or *that* must be silent in English.[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:

---

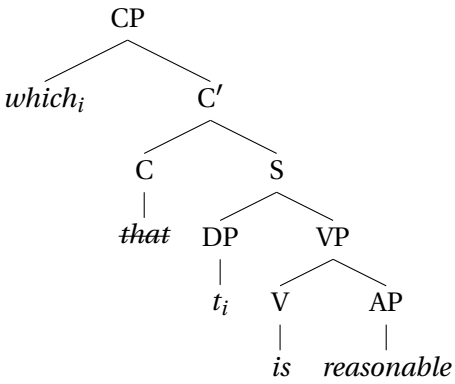
pronouns such as *who* or *which* is that only relative pronouns participate in so-called 'pied-piping':

(i) a. good old-fashioned values [$_{CP}$ on *which* we used to rely]
b. *good old-fashioned values [$_{CP}$ on *that* we used to rely]

This contrast can be understood under the assumption that *which* originates as the complement of *on*, and moves together with it, while *that* is generated in its surface position. Furthermore, the complementizer *that* is not found only in relative clauses; it also serves to introduce other finite clauses, as in *John thinks that Mary came*. Moreover, in some languages, including Bavarian German, relative pronouns can actually co-occur with complementizers (Carnie, 2013, Ch. 12).

[4] See Carnie (2013, Ch. 12) for a more thorough introduction to the syntax of relative clauses.

(42)



In this tree, the relative pronoun *which* is co-indexed with a trace in the subject position for the embedded auxiliary verb *is*. Because the movement changes only the underlying structure and not the sequence of words that is pronounced, this kind of movement is called STRING-VACUOUS MOVEMENT.

These syntactic assumptions lay the groundwork for a semantic treatment of relative clauses on which they function much like adjectival modifiers. The key assumptions are the following:

- Relative clauses are formed through a movement operation that leaves a trace.
- Traces are translated as variables.
- A relative clause is interpreted by introducing a lambda operator that binds this variable.

Which variable does a trace like t_3 correspond to? Recall that in L_λ we have an infinite number of constants and variables in stock. For every natural number i and every type τ, we have a constant of the form

$$c_{i,\tau}$$

and a variable of the form

$$v_{i,\tau}$$

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. Because e is in some sense the default type, we will sometimes abbreviate $v_{i,e}$ as v_i.

In the compositional system we are setting up here, a trace with a given index always will be translated as a variable with the same index. For example, the trace t_7 would be interpreted as v_7:

$$t_7 \rightsquigarrow v_7$$

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 v_7 will then depend on an assignment; recall from our definition of the semantics of variables in L_λ that:

$$[\![v_7]\!]^{M,g} = g(v_7)$$

Since traces are translated as variables, and variables are interpreted using assignment functions, traces ultimately get their denotation from assignment functions.[5]

We have thus arrived at a new composition rule:

Composition Rule 4. Pronouns and Traces Rule
If α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow v_i$

[5] Contrast Heim & Kratzer's (1998) rule, given in a direct interpretation style, where an assignment function decorates the denotation brackets: $[\![\alpha_i]\!]^g = g(i)$. Here the difference between direct and indirect interpretation becomes bigger than mere substitution of square brackets for squiggly arrows: In indirect interpretation, we translate pronouns and traces as logical variables. Note that the meta-language still contains its own variables in Heim and Kratzer's style, and these can be bound by lambda operators, as in $[\![\text{loves him}_i]\!]^g = \lambda x.\, x \text{ loves } g(i)$. Here, variables like x appear on the right-hand side and variables like 'him$_i$' appear on the left-hand side.

We have called it the 'Pronouns and Traces Rule' because it will also be used for pronouns; for example:

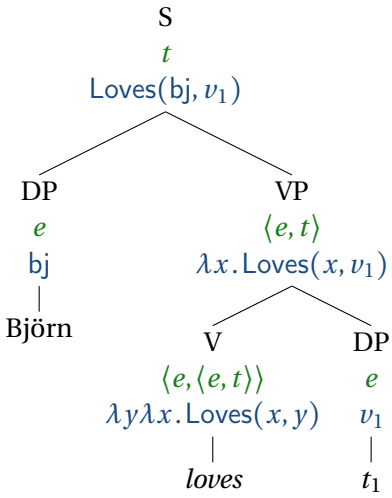
$$he_7 \rightsquigarrow v_7$$

We will see more on the pronoun side of this in Section 7.5. [6]

With these assumptions, we derive the representation

$$\mathsf{Loves}(\mathsf{bj}, v_1)$$

for *Björn loves* t_1:



The translation corresponding to this S node, $\mathsf{Loves}(\mathsf{bj}, v_1)$, 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 $\lambda x.\, \mathsf{Loves}(\mathsf{bj}, x)$?

[6] The idea of treating traces and pronouns as variables is rather controversial; see Jacobson (1999) and Jacobson (2000) for critique and alternatives.

We can achieve this by assigning the relative clause an interpretation in which a lambda operator binds the variable v_1, thus:

$$\lambda v_1 . \mathsf{Loves}(x, v_1)$$

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:

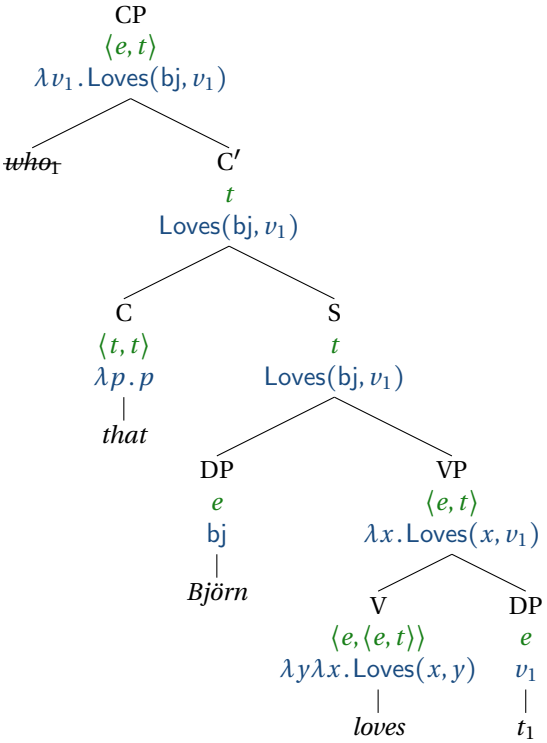
Composition Rule 5. Predicate Abstraction
If

- γ is a syntax tree whose only two subtrees are α_i and β
- $\beta \rightsquigarrow \beta'$
- β' is an expression of type t

Then $\gamma \rightsquigarrow \lambda v_i . \beta'$

This gives us the following analysis of the relative clause:

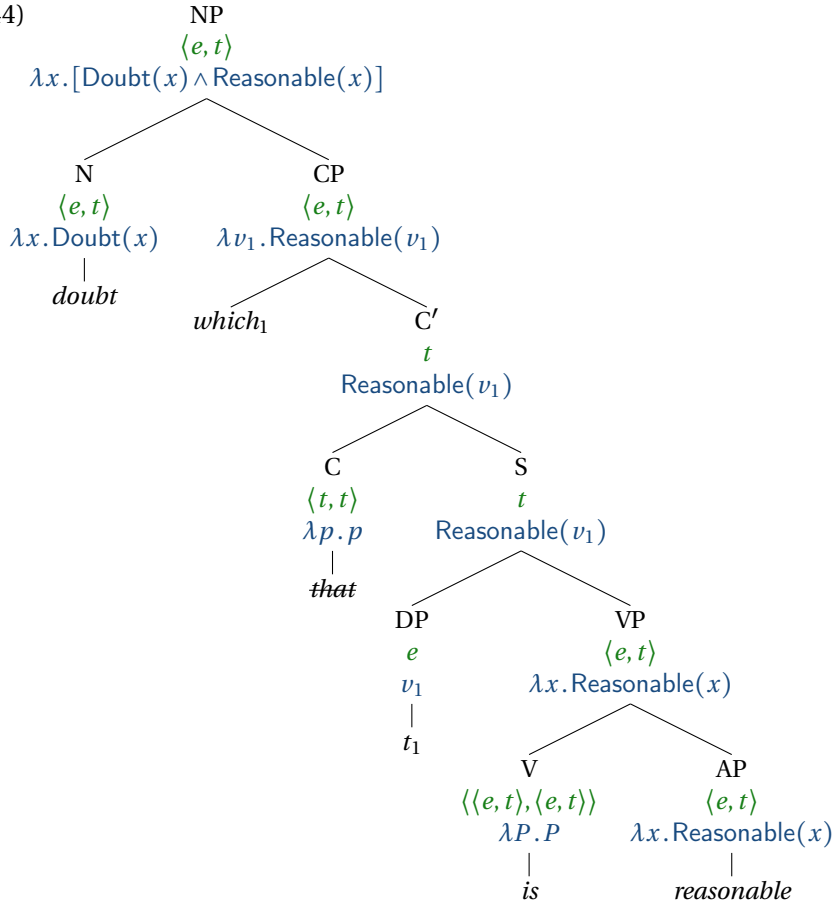
(43)



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.

Using the same tools, the phrase *doubt which is reasonable* can be given an analysis that accords with Judge Sweeney's intuitions:

(44)

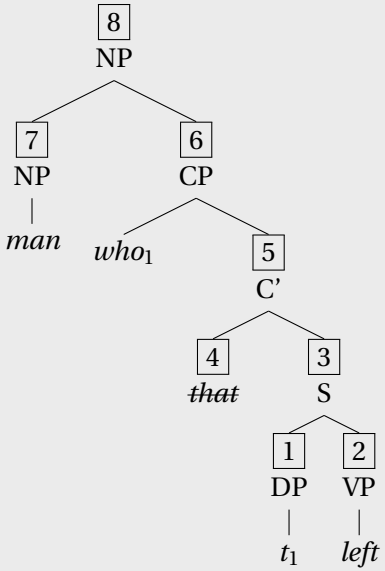


Under this analysis, a doubt which is reasonable is something that is both a doubt and reasonable.

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 (such as those where the complementizer is pronounced instead), as was illustrated in (43). 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 Abstrac-

tion, 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'.

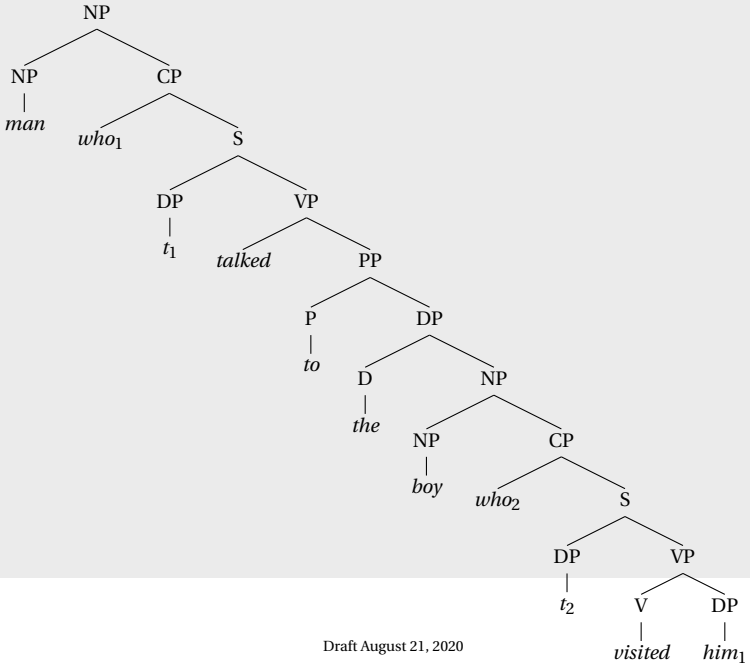
Exercise 9. Traditional grammar distinguishes between *restrictive* and *non-restrictive* relative clauses. Non-restrictive relative

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

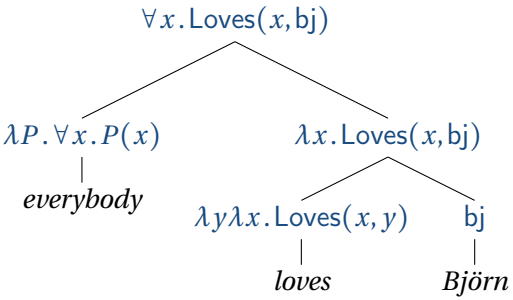
Everybody loves Björn should be translated as:

(45) $\forall x . \mathsf{Loves}(x, \mathsf{bj})$

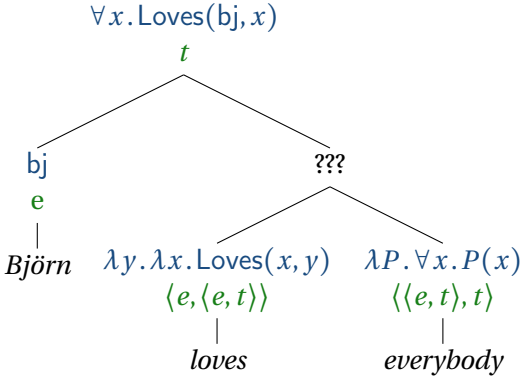
and *Björn loves everybody* should be translated as:

(46) $\forall x . \mathsf{Loves}(\mathsf{bj}, x)$

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:

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

The appropriate value for P here would be a function that holds of an individual if Björn loves that individual:

(48) $\lambda x.\mathsf{Loves}(\mathsf{bj}, x)$

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:

(49) $[\lambda P \forall x.P(x)](\lambda x.\mathsf{Loves}(\mathsf{bj}, x))$
$\equiv \forall x.\mathsf{Loves}(\mathsf{bj}, x)$

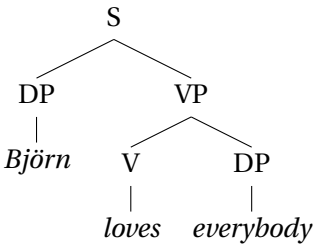
Exercise 11. Simplify the following expression step-by-step:

$$[\lambda Q.\forall x[\mathsf{Linguist}(x) \rightarrow Q(x)]](\lambda v_1.\mathsf{Offended}(\mathsf{j}, v_1))$$

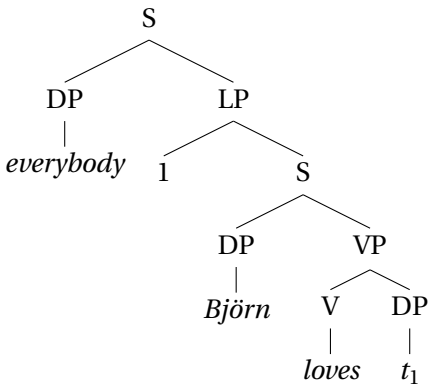
Tip: Use the 'scratch pad' function in the Lambda Calculator.

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). 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 (50a) is converted to the Logical Form representation (50b):

(50) a.



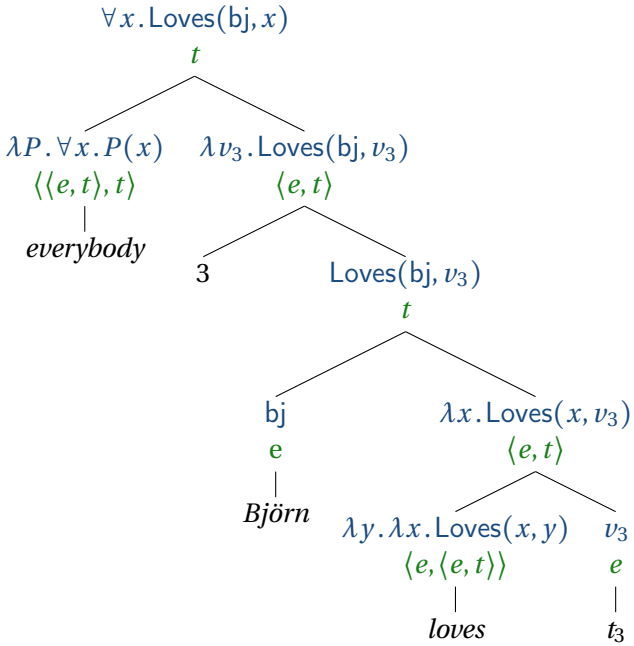
b.



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 a semantic fiction without 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.

(51)

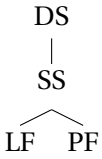


The Quantifier Raising solution to the problem of quantifiers in object position was originally developed in a syntactic theory with several levels of representation:

- 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. QR)[7]

Transformations map from DS to SS, and from SS to PF and LF:



This is the so-called 'T-model', or (inverted) 'Y-model' of Government and Binding theory, motivated originally by Wasow (1972) and Chomsky (1973). Since the transformations from SS to LF happen "after" the order of the words is determined, we do not see the output of these transformations. These movement operations are in this sense COVERT.

Many other transformational generative theories of grammar have been proposed over the years (see Lasnik & Lohndal 2013 for an overview), and many of these are also compatible with the idea of Quantifier Raising; the crucial thing is that there is an interface with semantics (such as LF) at which quantifiers are in the syntactic positions that correspond to their scope, and there is a trace indicating the argument position they correspond to. Quantifier Raising is not an option in non-transformational generative theories of grammar such as Head-Driven Phrase Structure Grammar (Pollard & Sag, 1994) and Lexical-Functional Grammar (Bresnan, 2001); other approaches to quantifier scope are taken in conjunction with those syntactic theories.

[7] '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 QR. 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 $\langle e, \langle e, t \rangle \rangle$. You can do this in the Lambda Calculator.

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. You can do this in the Lambda Calculator.

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, Terminal 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 struc-

tures which are then sent to the semantics for interpretation as a second step. 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.

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".

Alternatively, one can use type-shifting rules that have the power to repair the mismatch. These 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.

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.

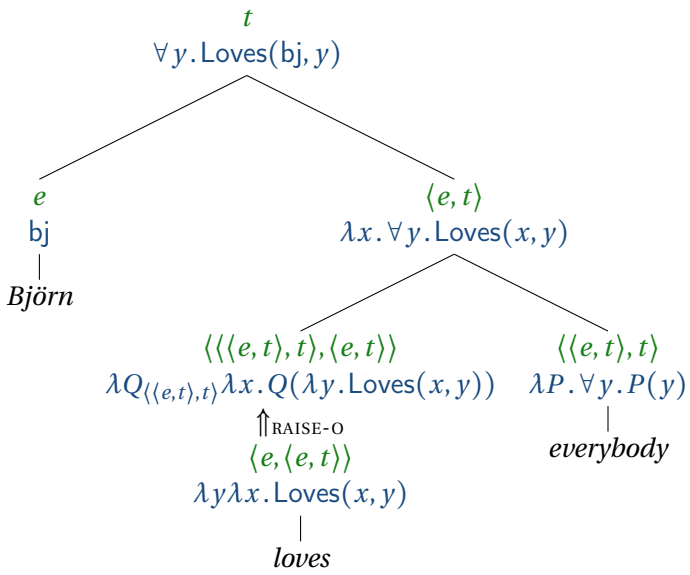
Type-Shifting Rule 2. Object raising (RAISE-O)
If an English expression α is translated into a logical expression α' of type $\langle e, \langle a, t\rangle\rangle$, for any type a, then α also has a translation of type $\langle\langle\langle e, t\rangle, t\rangle, \langle a, t\rangle\rangle$ of the 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:

(52)

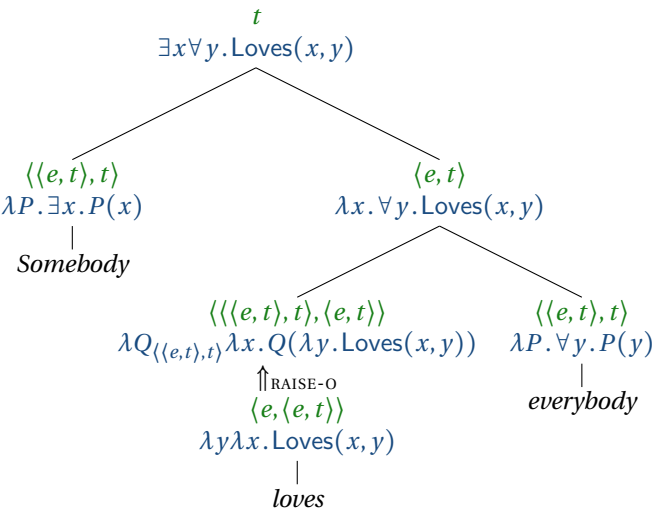


Exercise 16. Reproduce the tree in (52) using the Lambda Calculator, doing the beta-reductions along the way. Since the Lambda

Calculator does not support type-shifting, just treat the type shift as if it is a silent sister to *loves*. To represent the type variable a, write it as 'a.

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.

(53)



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 result-

ing 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:

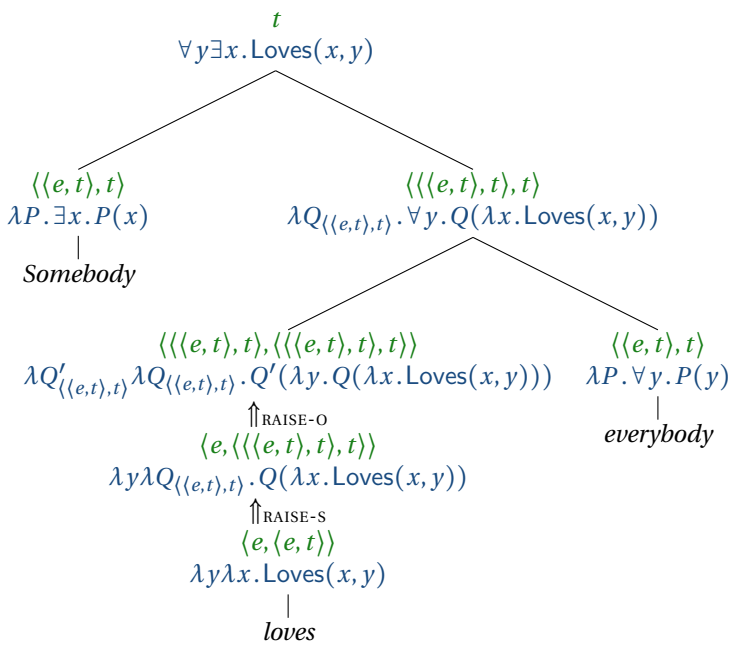
$$\lambda y_a \lambda Q_{\langle\langle e,t\rangle,t\rangle} . Q(\lambda x_e . \alpha'(y)(x))$$

(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:

(54)



Exercise 17. 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 (54). 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 general schema is as follows: If an expression has a translation α' of type $\langle \overrightarrow{a}, \langle b, \langle \overrightarrow{c}, t\rangle\rangle\rangle$, where $\overrightarrow{a}$ and $\overrightarrow{c}$ are possibly null sequences of types, then that expression also has translations of the following form, where $\overrightarrow{x}$ and $\overrightarrow{z}$ stand for possibly null sequences of arguments of the same length as $\overrightarrow{a}$ and $\overrightarrow{c}$ respectively:

$$\lambda \overrightarrow{x}_{\overrightarrow{a}} \lambda Q_{\langle\langle b,t\rangle,t\rangle} \lambda \overrightarrow{z}_{\overrightarrow{c}} [Q(\lambda y_b[\alpha(\overrightarrow{x})(y)(\overrightarrow{z})])]$$

(unless x, y, z, or Q occur in α'; in that case, just use different variables of the same type).

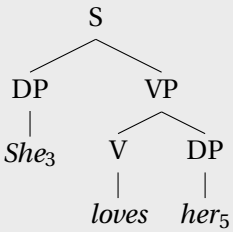
This schema works in the following way, for a verb α that expects at least one argument, the "targeted argument" as we will call it. In the following examples, this argument will be of type e, but more generally it could be of any type; this is why the schema uses b instead of e. The sequences $\overrightarrow{x}$ 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 give a concrete example, the Object Raising rule above results from applying this 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). 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.

7.5 Pronouns

Recall that the Pronouns and Traces Rule tells us that if α is an indexed trace or pronoun, $\alpha_i \rightsquigarrow v_i$. Thus pronouns and traces are interpreted in the same manner: as variables. In this section, we will try and justify this assumption.

Exercise 18. 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:

(55) She is suspicious.

then this occurrence of *she* would refer to Cruella De Vil. But I could point to Ursula and say the same thing, in which case *she* would refer to Ursula. I don'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, I could raise Ursula to salience by talking about her:

(56) 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:

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?"

(57) No woman blamed herself.

(58) Neither man thought he was at fault.

(59) Every boy loves his mother.

(60) the book such$_1$ that Mary reviewed it$_1$

So not all pronouns are referential. It is sometimes said that *No woman* and *herself* are "coreferential" in (57) but this is strictly speaking a misuse of the term "coreferential", because, as **?** point out, *coreference implies reference*.

Exercise 19. 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 (57)-(60) can be analyzed as bound variables. For example, (57) should be translated as:

(61) $\neg\exists x.[\mathsf{woman}(x) \wedge \mathsf{blamed}(x,x)]$

Furthemore, there are certain cases where pronouns behave almost identically to traces. For instance, regarding U.S. Supreme Court Justice Ruth Bader Ginzburg, it was once remarked:

(62) This is an older woman who everyone listens when **she** speaks.

The alternative with an unpronounced trace (**...who everyone listens when speaks*) would have been ungrammatical; inserting the pronoun *she* rescues the sentence. 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 when she speaks

denotes the property of being an x such that everyone listens when x speaks.

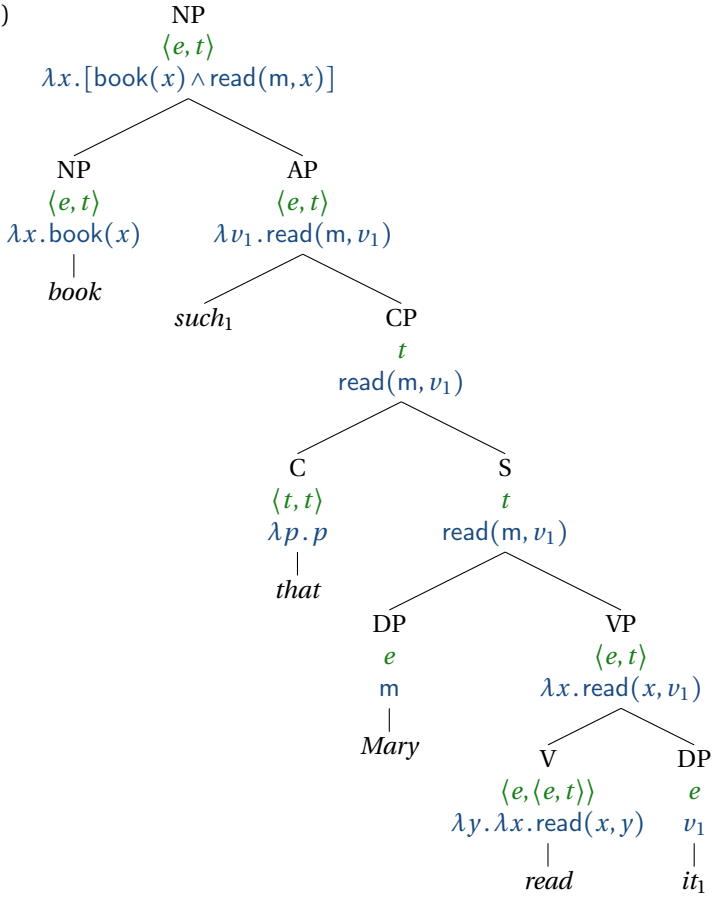
Another example in which pronouns are interpreted very much like traces is with *such*-relatives, as in:

(63) the book such that Mary read it

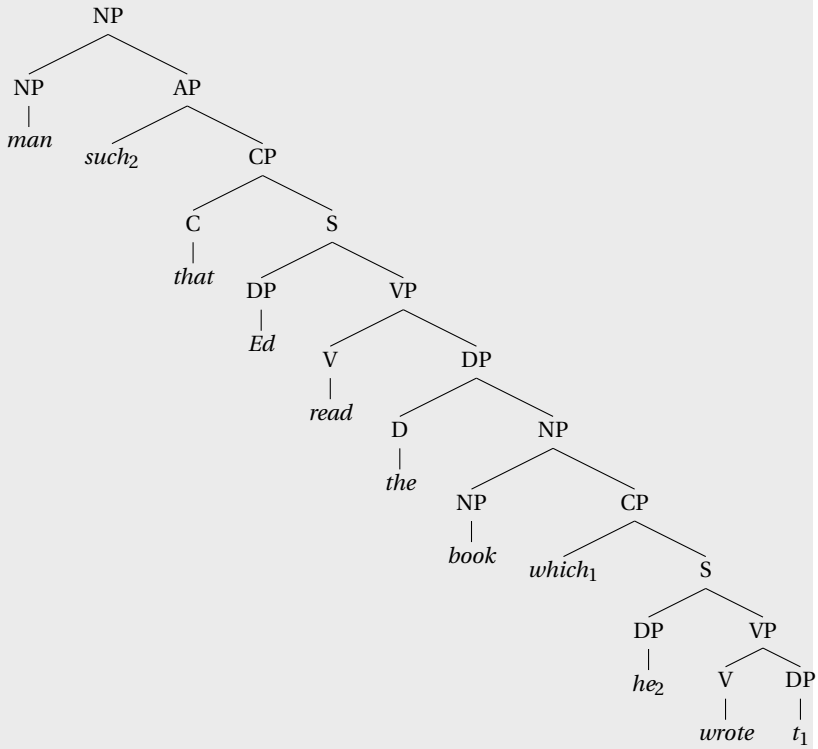
These cases can be treated much like relative clauses, using Predicate Abstraction. The trigger for the abstraction in this case is *such*, which is coindexed with a pronoun rather than a trace. For example, in (60), there is coindexation between *such* and *it*. The analysis works as follows:[8]

[8] Let us assume that x is a distinct variable from v_1.

(64)



Exercise 20. Give the types and a fully beta-reduced logical translation for every node of the following tree.



For the definite article, assume the treatment described in Exercise 10.

In light of this evidence, let us consider the possibility that pronouns should *always* be treated as bound variables.[9]

[9]Heim & Kratzer (1998, pp. 116-118) define a distinction between free and bound variables in natural language in their 'direct interpretation' framework, where the interpretation is relative to an assignment function that applies directly to pronouns and traces, rather than their interpretations. Because we are using 'indirect interpretation', we can let the notions of 'free' and 'bound' for natural language expressions be inherited from their standard conceptions in logic.

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. Currently, Lambda Abstraction is the only mechanism by which variables get bound, so if this condition holds, then the pronoun is part of an expression that translates as a lambda abstraction binding the logical variable corresponding to the pronoun, so there is a corresponding kind of binding at the logical level.

Hypothesis 2 has a number of undesirable consequences. It would mean that for cases like (56), we would have to QR *Ursula* to a position where it QRs *She* in the second sentence somehow. It is not completely crazy to imagine that proper names can undergo QR, but it is widely assumed that QR does not allow a DP to move across a sentence boundary. If that were possible, then we would get all sorts of interpretations for quantifiers that we never get. For example, we should get a reading where *everybody* binds *he* in the sequence, *I don't think everybody should be invited. He will just be a bore.*

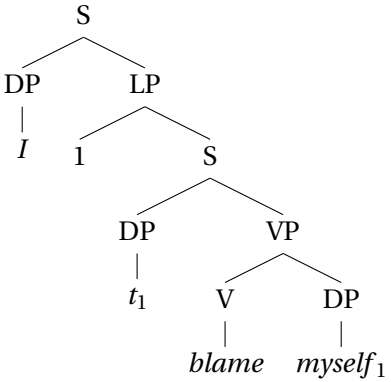
Moreover, we don't want to treat all pronouns as bound variables because there are some ambiguities that depend on a distinction between free and bound interpretations. For example, in the movie *Ghostbusters,* there is a scene in which the three Ghostbusters Dr. Peter Venkman, Dr. Raymond Stanz, and Dr. Egon Spengler (played by Bill Murray, Dan Akroyd, 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.

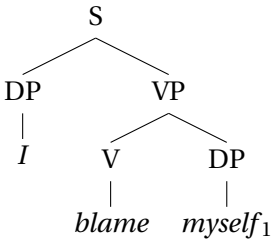
There are two readings of Peter Venkman's quip, a sympathetic reading and a reading on which he is as usual being a jerk. On the SLOPPY reading (the sympathetic reading), Peter blames himself. On the STRICT reading (the asshole reading), Peter blames Egon. The strict/sloppy ambiguity exemplified in (65) 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 QR thus:

(66)



The strict reading can be derived from an antecedent without QR:

(67)



This suggests that pronouns are sometimes bound, and sometimes free. Note, however, we have not said anything about how to interpret deictic pronouns like *I*, nor how we might ensure that

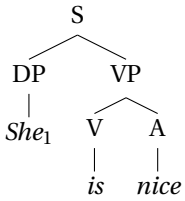
myself is interpreted as co-referential with *I*, so a full explanation of this contrast awaits an answer to these questions.

Exercise 21. Which reading — strict or sloppy — involves a bound interpretation of the pronoun? Which reading involves a free interpretation?

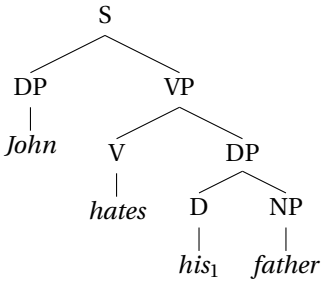
Heim and Kratzer's hypothesis: All pronouns are variables, and bound pronouns are interpreted as bound variables, and referential pronouns are interpreted as free variables.

In the following examples, the pronoun in the sentence is free:

(68)

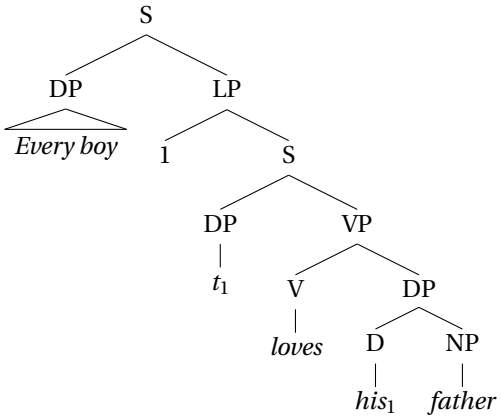


(69)

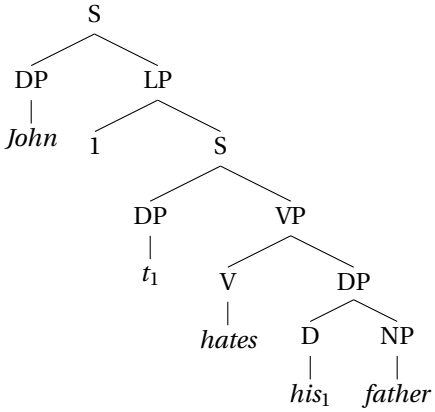


But in these examples, the pronoun is bound:

(70)



(71)



Whether or not QR 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 22. 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 func-

tions can be thought of as being provided by the discourse context. As Heim & Kratzer (1998) put it:

> Treating referring pronouns as free variables implies a new way of looking at the role of variable assignments. Until now we have assumed that an LF whose truth-value varied from one assignment to the next could *ipso facto* not represent a felicitous, complete utterance. We will no longer make this assumption. Instead, let us think of assignments as representing the contribution of the utterance situation.

Still, it is not appropriate to say *She left!* if your interlocutor has no idea who *she* refers to. We can capture this idea using dynamic semantics, discussed in Chapter 9.

Exercise 23. 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, (72) and (73) differ in voice-marking and are true in different circumstances. (The morphological glosses have been simplified.)

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

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

In (72), 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 7.1, but not in the situation on the left. (Images created by Benjamin Bruening for the Scope Fieldwork Project; see `http://udel.edu/~bruening/scopeproject/materials.html`.)

In (73), 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 7.1 makes it true, and one where the picture on the right makes it true.

(a) Write out representations in L_1 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?

Figure 7.1: 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 (73).)

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:

(4) Not every dubstep album by Frege is famous.

Thus neither the original sentence nor its negation is felt to be true. How can this be?

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) Is every dubstep album by Frege famous?

(6) If every dubstep album by Frege is famous, then I must be out of the loop.

Every member of this FAMILY OF SENTENCES shares the implication; this is characteristic of presupposition.

A word or construction that signals a presupposition is called a PRESUPPOSITION TRIGGER. Other presupposition triggers include the quantifiers *both* and *neither*, factive adjectives like *glad*, factive verbs like *know*, possessives, exclusives like *only*, and the definite determiner *the* (also called a definite article). Besides *every*, here are some examples (where ≫ signifies 'presupposes'):

(7) a. Neither candidate is qualified.
≫ There are exactly two candidates.
b. Ed is glad we won.
≫ We won.

c. Ed knows we won.
   ≫ We won.
d. Ed's son is bald.
   ≫ Ed has a son.
e. Only Ed came.
   ≫ Ed came.
f. The balcony is lovely.
   ≫ There is 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 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.

Let us first observe that definite descriptions often convey uniqueness. 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 I were to tell you: *Guess what! I'm having dinner with the princess tonight.* You would probably infer that there is a princess, and that there is only one. Thus definite descriptions convey EXISTENCE (that there is a 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 have just analyzed. He proposes that *The princess smokes* means 'There is exactly one princess and she smokes':

$$\exists x.[\mathsf{Princess}(x) \wedge \forall y.[\mathsf{Princess}(y) \rightarrow x = y] \wedge \mathsf{Smokes}(x)]$$

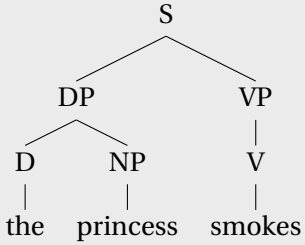
According to Russell's treatment, the definite determiner introduces a logical entailment both that there is a princess (existence) and that there is only one (uniqueness). This means that the sentence is predicted to be false if there are either no princesses or multiple ones.

Exercise 1. Read the above formula aloud to yourself and then write out the words that you said. Which part of this formula ensures uniqueness?

Exercise 2.

(a) Give a Russellian lexical entry for *the*. What is the type of *the* under your treatment?

(b) Show this lexical entry in action in the following tree:



Strawson (1950), in a response to Russell titled "On Referring", 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 ask you whether you thought that what he had just said was true, or was false; whether you agreed or disagreed with what he had just said. I think you would be inclined, with some hesitation, to say that you did not do either; that the question of whether his statement

[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.

> was true or false simply *did not arise*, because there was no such person as the King of France. You might, if he were obviously serious (had a dazed, astray-in-the-centuries look), say something like: *I'm afraid you must be under a misapprehension. France is not a monarchy. There is no King of France.*

Strawson's observation is that we feel squeamish when asked to judge whether a sentence of the form *The F is G* is true or false, when there is no F. We do not feel that the sentence is false; we feel that the question of its truth does not arise, as Strawson put it.

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. In fact, one way of defining presupposition is just in this way:

(8) **Semantic definition of presupposition**
A presupposes B if and only if:
Whenever A is true or false, B is true.

Exercise 3. 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. Is semantic presupposition a species of entailment? Or is it the other way around? Or neither? Explain your reasoning.

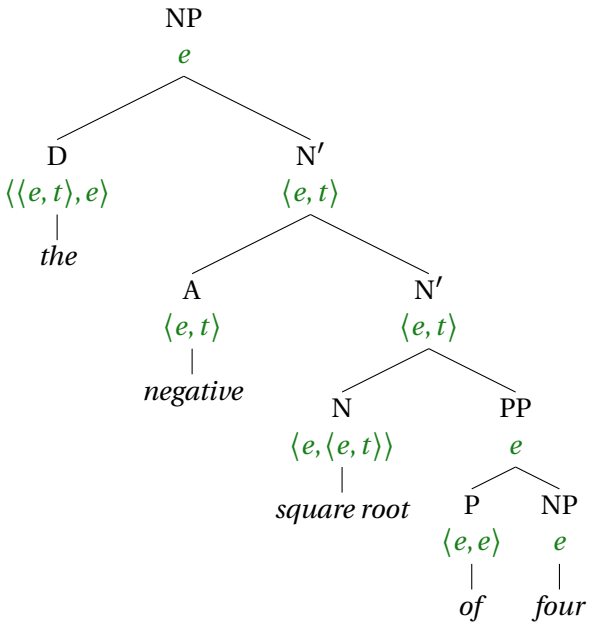
One way of implementing the idea that sentences might be neither true nor false is by introducing a third truth value. Under this strategy, along with 'true' and 'false', we have 'undefined'

or 'nonsense' as a truth value. It turns out that having this third truth value makes the formal system a bit easier to set up, so we will adopt that strategy here. Let us use # 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 bald* will be #. 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. This intuition was expressed earlier by Frege (1892 [reprinted 1948]). According to Frege, a definite description like *the king of France* or *the negative square root of 4*, 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, Heim & Kratzer (1998) suggest that *square root* is a "transitive noun", with a denotation of type $\langle e, \langle e, t\rangle\rangle$, and that "*of* is vacuous, *square root* applies to 4 via Function Application, and the result of that composes with *negative* under predicate modification." Spelling this out yields the following structure:



Now, 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 simply does not return a value.

Another 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. We are not adding this symbol to our logical representation language L_λ; 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

[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'.

of the form *the F* will denote $\#_e$ whenever the number of satisfiers of *F* is not exactly one.

To formalize this idea, we introduce a new symbol into our logic:

$$\iota$$

which is the Greek letter 'iota'. Like the λ symbol, ι can bind a variable. Here is an example:

$$\iota x.P(x)$$

This is an expression of type e. It denotes the unique individual satisfying P if there is exactly one such individual, otherwise it denotes $\#_e$. To add this symbol to our logic, first we add a syntax rule producing ι-expressions:

Syntax rule: Iota
If ϕ is an expression of type t, and u is a variable of type e, then $\iota u.\phi$ is an expression of type e.

The semantics of iota-expressions is defined as follows:

Semantic rule: Iota
$$[\![\iota u.\phi]\!]^{M,g} = \begin{cases} d \text{ if } \{k \mid [\![\phi]\!]^{M,g[u\mapsto k]} = 1\} = \{d\} \\ \#_e \text{ otherwise} \end{cases}$$

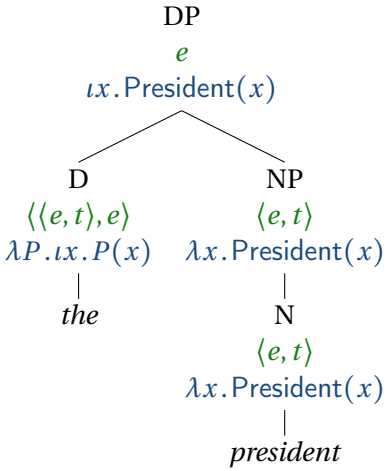
Exercise 4. Read the semantic rule for ι aloud to yourself and then write down the words that you said. How does this definition ensure that ι expressions are undefined when existence and uniqueness are not satisfied?

With this formal tool in hand, we can now give a Fregean analysis of the definite determiner as follows:

(9) $the \rightsquigarrow \lambda P.\iota x.P(x)$

Applied to a predicate-denoting expression like $\lambda x.\mathsf{President}(x)$, it denotes the unique president, if there is one and only one president in the relevant domain.

(10)



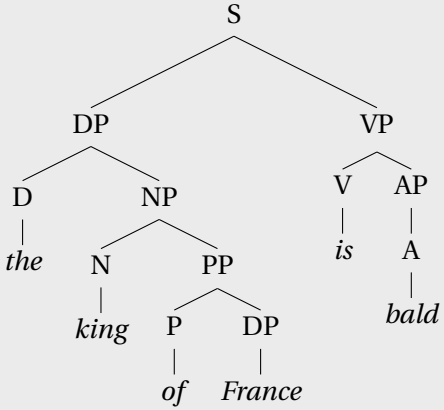
(11) $[\![\iota x.\mathsf{President}(x)]\!]^{M,g}$
$= \begin{cases} d \text{ if } \{k : [\![\mathsf{President}(x)]\!]^{M,g[x\mapsto k]} = 1\} = \{d\} \\ \#_e \text{ otherwise} \end{cases}$

For example, in a model corresponding to the White House in 2014, where the only president in the relevant domain is Barack Obama, the denotation of the phrase would be Barack Obama.

Now, let us assume that a predicate like Bald yields the undefined truth value # when given the undefined individual $\#_e$ as input, and yields true or false only for other individuals. So $[\![\mathsf{Bald}(\alpha)]\!]^{M,g}$ will be # if $[\![\alpha]\!]^{M,g} = \#_e$, and 1 or 0 otherwise, depending on whether $[\![\alpha]\!]^{M,g}$ is in the extension of Bald with respect to M and g. 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 bald* would then have # as its denotation.

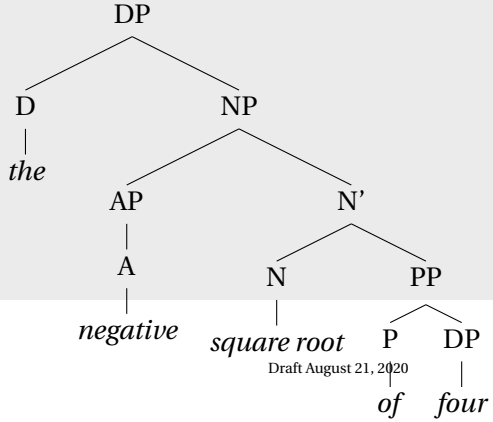
Exercise 5. Explain how this Fregean treatment of the definite determiner vindicates Strawson's intuitions.

Exercise 6. Using the assumptions above, compute a derivation for the following tree:



This exercise can be solved using the Lambda Calculator.

Exercise 7. 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:

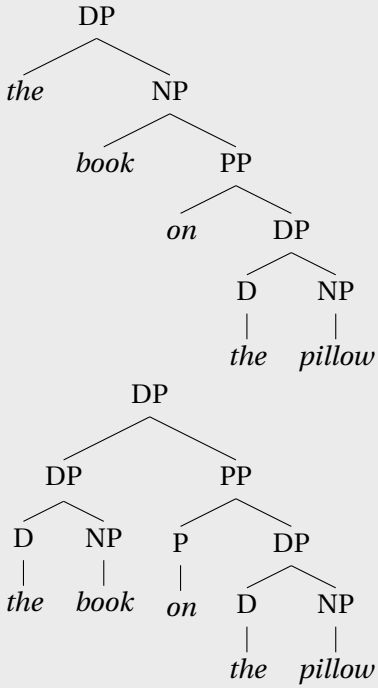


This exercise can be solved using the Lambda Calculator.

Exercise 8. Assume the following lexical entries:

1. *book* $\rightsquigarrow \lambda x.\mathsf{Book}(x)$
2. *on* $\rightsquigarrow \lambda y \lambda x.\mathsf{On}(x, y)$
3. *pillow* $\rightsquigarrow \lambda x.\mathsf{Pillow}(x)$

Which of the following trees 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.

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:

(12) 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 Bald) 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 9. Both *The king of France is bald* and *The opera by Mozart is Italian* have an undefined value relative to the actual world, but for different reasons. Explain the difference.

8.3 Definedness conditions

In the previous section, we encountered one example of a presuppositional expression, namely the definite determiner *the*. We translated the definite determiner 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 $\#_e$.

The definite determiner is one of many presupposition triggers. Even if we are satisfied with our treatment of it, we still need a more general way of dealing with presupposition. The determiners *both* and *neither*, for example, come with presuppositions. In a context where three candidates are applying for a job, it would be quite odd for someone to say either of the following:

(13) a. Both candidates are qualified.
b. Neither candidate is qualified.

If there were only two candidates and both were qualified, then (13a) would clearly be true and (13b) would clearly be false. But with any number of candidates other than two, it is odd to say that these sentences are true or false. This suggests 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*. We can model this 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 way of expressing the fact that predicate P has exactly two satisfiers.[3] This is what is presupposed. To signify that it is presupposed, we will use Beaver & Krahmer's (2001) ∂ 'partial' operator. 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:

(14) $\textit{neither} \rightsquigarrow \lambda P \lambda Q . [\partial(|P| = 2) \wedge \neg \exists x . [P(x) \wedge Q(x)]]$

This says that *neither* is basically a synonym of *no*, carrying an extra presupposition: that there are exactly two Ps.

In order to be able to give translations like this, we need to

[3] $|P| = 2$ is short for $\exists x \exists y [\neg(x = y) \wedge P(x) \wedge P(y) \wedge \neg \exists z [\neg(z \neq x) \wedge \neg(z = y) \wedge P(z)]]$.

augment L_λ to handle formulas containing the ∂ symbol. Let us call our new language ∂L. In this new language, $\partial(\phi)$ will be an kind of expression of type t. Its value will be 'true' if ϕ is true and 'undefined' otherwise. To implement this, we must add the following clause to our syntax rules:

Syntax Rule: Definedness conditions
If ϕ is an expression of type t, then $\partial(\phi)$ is an expression of type t.

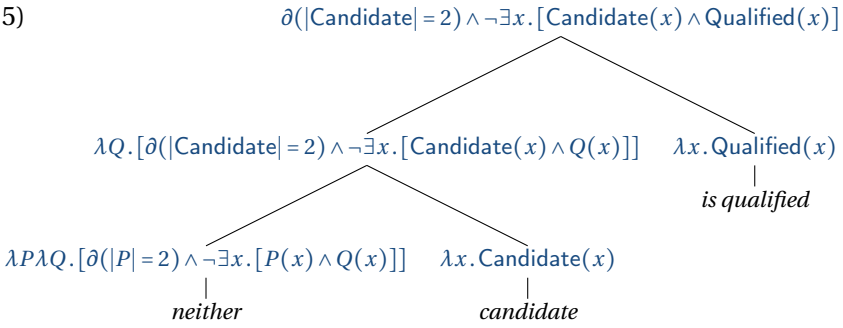
We define the semantics of these expressions as follows:

Semantic Rule: Definedness conditions
If ϕ is an expression of type t, then:

$$[\![\partial(\phi)]\!]^{M,g} = \begin{cases} 1 \text{ if } [\![\phi]\!]^{M,g} = 1 \\ \# \text{ otherwise.} \end{cases}$$

The lexical entry in (14) will give us the following analysis for (13b), where beta-reduced variants of the translations are given at each node:

(15)



The translation for the whole sentence should have a defined value in a model if |Candidate| = 2 is true in the model. If it has a de-

$\wedge$	T	F	#
T	T	F	#
F	F	F	#
#	#	#	#

$\vee$	T	F	#
T	T	T	#
F	T	F	#
#	#	#	#

	$\neg$
T	F
F	T
#	#

	∂
T	T
F	#
#	#

Table 8.1: Truth tables for the Weak Kleene connectives

fined value, then its value is equal to that of $\neg\exists x.[\text{Candidate}(x) \rightarrow \text{Qualified}(x)]$.[4]

The existence presupposition of the quantifier *every* can be treated using definedness conditions as well. We can capture it using the following kind of analysis of *every*:

(16) $\quad \textit{every} \rightsquigarrow \lambda P \lambda Q.[\partial(\exists x.P(x)) \wedge \forall x.[P(x) \rightarrow Q(x)]]$

This 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.

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 leads to the truth tables in Table 8.1. In the truth tables for the binary connectives, the truth value of the left-hand conjunct (or disjunct) is represented by the row labels and the truth value of the right-

[4] Beta-reduction works as usual under this way of doing things. Although the notation here is quite similar to Heim & Kratzer's (1998) notation for partial functions, and the expressive power is the same, the style of handling undefinedness laid out in Heim & Kratzer (1998) does not allow for beta-reduction to work properly, so the present system is a bit cleaner in this respect.

$\wedge$	T	F	#
T	T	F	#
F	F	F	**F**
#	#	**F**	#

$\vee$	T	F	#
T	T	T	**T**
F	T	F	#
#	**T**	#	#

	$\neg$
T	F
F	T
#	#

	∂
T	T
F	#
#	#

Table 8.2: Truth tables for the Strong Kleene connectives

hand conjunct (or disjunct) is represented by the column labels. The value in the table is the value for the conjoined (or disjoined) formula. These connectives are called the WEAK KLEENE connectives.

If, on the other hand, we take undefinedness 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).

Now, what happens when a universal quantifier takes scope over a presupposition operator? Consider the following example:

(17) Every boy loves his cat.

This would be translated:

(18) $\forall x.[\mathsf{Boy}(x) \rightarrow \mathsf{Loves}(x, \iota y[\mathsf{Cat}(y) \wedge \mathsf{Has}(x,y)])]$

This formula will be true in a model where every element of D_e satisfies the formula:

$$[\mathsf{Boy}(x) \rightarrow \mathsf{Loves}(x, \iota y[\mathsf{Cat}(y) \wedge \mathsf{Has}(x,y)])]$$

This formula will denote the value 'undefined' when x is a boy who doesn't happen to have a cat. What if there are such boys? Should that make the sentence as a whole have an undefined truth value? Or should we say that the sentence is true as long as every boy *who has a cat* loves it? In other words, if the assortment of

truth values that the proposition *x loves his cat* (the SCOPE PROPOSITION) takes on as we cycle through various values for x contains both 1 and #, should the truth value for the proposition *Every boy loves his cat* (the UNIVERSAL PROPOSITION) be 1 or should it be #? Different authors have advocated different answers to this question. LaPierre (1992) and Muskens (1995a) assign the value # to a universal claim consisting of 1s and #s. Haug (2013) allows a universal claim to be true as long as it contains no 0s, unless it is *always* undefined.

As Muskens (1995a) discusses, however we set things up, it should be the case that our universal quantifiers 'match' our treatment of conjunction, and our existential quantifiers 'match' our treatment of disjunction. So a universal claim should be seen as a big conjunction, and an existential claim should be seen as a big disjunction. This leads to the following treatment of universal quantification:

$$[\![\forall x.\phi]\!]^{M,g} = \begin{cases} 1 & \text{if } [\![\phi]\!]^{M,g[x\to k]} = 1 \text{ for all } k \in D \\ \# & \text{if } [\![\phi]\!]^{M,g[x\to k]} = \# \text{ for some } k \in D \\ 0 & \text{otherwise} \end{cases}$$

So a universal claim is false only if the scope proposition never takes on an undefined value, and is not always true. For example, *Every boy loves his cat* is false only if every boy has a cat (and at least one boy doesn't love his cat).

If we want to maintain that $\forall x.\phi$ is equivalent to $\neg\exists x.\neg\phi$, then this yields the following interpretation of the existential quantifier:

$$[\![\exists x.\phi]\!]^{M,g} = \begin{cases} 0 & \text{if } [\![\phi]\!]^{M,g[x\to k]} = 0 \text{ for all } k \in D \\ \# & \text{if } [\![\phi]\!]^{M,g[x\to k]} = \# \text{ for some } k \in D \\ 1 & \text{otherwise} \end{cases}$$

So an existential claim is true only if the scope proposition never takes on an undefined value, and is not always false. For example,

Some boy loves his cat is false if every boy has a cat (and no boy loves his cat).

Should the undefined individual be considered part of D? Let's assume not, because if we did, then too many universal and existential claims would turn out to be undefined.

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 the undefined individual? 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. To deal with this issue, LaPierre (1992) defines identity between two terms as follows:

- If neither α nor β denotes the undefined individual, then $[\![\alpha = \beta]\!]^{M,g} = 1$ if $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$, and 0 otherwise.
- If one of α or β denotes the undefined individual, then $[\![\alpha = \beta]\!]^{M,g} = 0$
- If both denote the undefined individual, then $[\![\alpha = \beta]\!]^{M,g} = \#$ is undefined (the rationale being that not enough is "known" about the objects to determine that they are the same or distinct).

This treatment avoids the conclusion that *The King of France is Grand Sultan of Germany* is true.

Now, generally, predications involving the undefined individual will themselves be undefined. But there is one case in which we might want the predication to be true. Consider the following sentence discussed by Russell (1905):

(19) The golden mountain does not exist.

One possible treatment of this sentence is with an existence predicate, which is true of actual individuals and false of undefined individuals. Let us name this predicate Exists. Then the translation of (19) would be:

(20) $\neg\mathsf{Exists}(\iota x.[\mathsf{Golden}(x) \wedge \mathsf{Mountain}(x)])$

(This is not the only possible treatment of this example.) The semantics of Exists might then be defined as follows:

Semantic rule: Existence predicate
$[\![\mathsf{Exists}(\alpha)]\!]^{M,g} = 1$ if $[\![\alpha]\!]^{M,g} \neq \#_e$ and 0 otherwise

Assuming that $\iota x.[\mathsf{Golden}(x) \wedge \mathsf{Mountain}(x)])$ denotes $\#_e$, the sentence is correctly predicted to be true under this treatment.

We are now ready to give the full semantics for ∂L. We leave the syntax of the language implicit, and just give the semantics here.

As usual, types are associated with domains. Type e is associated with the domain of individuals $D_e = D$ and type t is associated with the domain of truth values $D_t = \{1, 0, \#\}$. For functional types $\langle\sigma,\tau\rangle$, there is a domain $D_{\langle\sigma,\tau\rangle}$ consisting of the (total) functions from D_σ to D_τ. For every type, there is also an 'undefined individual' of that type, which we refer to as $\#_\tau$.

Expressions are interpreted with respect to a model, a world, and an assignment. A model is a tuple $\langle D, I\rangle$ subject to the following constraints:

- The domain of individuals D_e contains at least one individual.
- I is an interpretation function, assigning a denotation to all of the constants of the language. The denotation of a constant of type τ is a member of D_τ.

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 $g(u) \in D_\tau$. We use $g[x \to d]$ to denote an assignment function which is exactly like g with the possible exception that $g(x) = d$.

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. **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. **Equality**
If α and β are terms, then
$$[\![\alpha = \beta]\!]^{M,g} = \begin{cases} 1 & \text{if } [\![\alpha]\!]^{M,g} \neq \#_e \text{ and } [\![\beta]\!]^{M,g} \neq \#_e \text{ and } [\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g} \\ \# & \text{if } [\![\alpha]\!]^{M,g} = \#_e \text{ and } [\![\beta]\!]^{M,g} = \#_e \\ 0 & \text{otherwise} \end{cases}$$

4. **Connectives**
The semantics of the connectives is defined as in Table 8.1.

5. **Quantification**

(a) If ϕ is a formula and v is a variable of any type: then
$$[\![\forall v\phi]\!]^{M,g} = \begin{cases} 1 & \text{if } [\![\phi]\!]^{M,g[v\to k]} = 1 \text{ for all } k \in D \\ \# & \text{if } [\![\phi]\!]^{M,g[v\to k]} = \# \text{ for some } k \in D \\ 0 & \text{otherwise} \end{cases}$$

(b) If ϕ is a formula and v is a variable of any type: then
$$[\![\exists v\phi]\!]^{M,g} = \begin{cases} 0 & \text{if } [\![\phi]\!]^{M,g[v\to k]} = 0 \text{ for all } k \in D \\ \# & \text{if } [\![\phi]\!]^{M,g[v\to k]} = \# \text{ for some } k \in D \\ 1 & \text{otherwise} \end{cases}$$

6. **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.

8.4 The projection problem

The treatment of presupposition we have given so far captures PRESUPPOSITION PROJECTION. If there are no candidates, then *Every candidate is qualified* has no truth value, nor does *It is not the case that every candidate is qualified.* Both imply —in some sense of *imply*— that there are candidates. In that sense, the presupposition PROJECTS over negation. The presupposition also projects from the antecedent of a conditional: *If every candidate is qualified, then it doesn't matter who we pick* also, as a whole, communicates that the speaker takes there to be at least two candidates, as does the question, *Is every candidate qualified?*

But presuppositions do not always project. Consider the following examples:

(21) If there is a king of France, then the king of France is wise.

(22) 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 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*. The problem of determining when a presupposition projects is called the PROJECTION PROBLEM.

Operators like *if*/*then* and *either*/*or* do let some presuppositions through, for example:

(23) If France remains neutral, then the king of France is wise.

(24) Either France is lucky or the king of France is wise.

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.

In (21), for example, the consequent (*the king of France is wise*) presupposes that there is a king of France, and the antecedent of the conditional is *there is a king of France*. The antecedent entails of course that there is a king of France, so the presupposition gets filtered out. In (23), the antecedent is *France remains neutral*, which doesn't entail that there is a king of France, so the presupposition "passes through the filter", so to speak.

With a disjunction, the generalization is that presupposition of one disjunct gets filtered out when the negation of another disjunct entails it.

In (22), for example, the second disjunct (*the king of France is wise*) presupposes that there is a king of France. The first disjunct is *there is no king of France*, whose negation is *there is a king of France*, which again entails of course that there is a king of France, so the presupposition gets filtered out. In (24), the first disjunct does not entail that there is a king of France, so the presupposition does not get filtered out.

We have used the word *entail* in the generalizations above. In (21) and (22), 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 (example adapted from Karttunen 1973a:

(25) Either Geraldine is not a catholic 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 local context for the second disjunct is the negation of the first disjunct. The first disjunct is *Geraldine is not a catholic*, so the local context is *Geraldine is a catholic*. If we assume that all catholics used to attend services on Sundays (and most still do), then the antecedent entails that Geraldine attends services on Sundays. So the generalization above correctly captures the fact that (25) does not presuppose that Geraldine attends services on Sundays.

The system that we have introduced for dealing with presuppositions might seem to predict that presuppositions will always project, since undefinedness tends to "percolate up," so to speak. The projection problem has been dealt with using *dynamic semantics*, where the denotation of a sentence is a "context change potential": a function that can update a discourse context. We take up this subject in the next chapter.

9 Dynamic semantics

(Significant revisions are planned for this chapter.)

9.1 Introduction

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. We will show first that the presupposition projection problem receives an insightful solution under a dynamic perspective. We then discuss pronouns with indefinite antecedents, including the famous 'donkey sentences':

(1) If a farmer owns a donkey, then he beats it.

The chapter ends with a compositional dynamic fragment.

9.2 Presupposition in dynamic semantics

Recall the following generalizations from the previous chapter:

(2) When the antecedent of the conditional (the *if*-part) entails a presupposition of the consequent (the *then*-part), the presupposition gets filtered out.

[1] Heim 1982b, 1983b,a; Kamp & Reyle 1993; Groenendijk & Stokhof 1990a, 1991; Muskens 1996, among others

(3) A presupposition of one disjunct gets filtered out when the negation of another disjunct entails it.

These two generalizations can be stated concisely and illuminatingly using Karttunen's (1974) concept of LOCAL CONTEXT: In general, a presupposition gets filtered out if it is entailed by the appropriate local context. The local context for the consequent of a conditional is its antecedent, and the local context for one disjunct of a disjunction is the negation of the other disjunct.

This idea builds on Stalnaker's (1978) ideas about the pragmatics of presupposition. Stalnaker introduces the concept of the CONTEXT SET, which is conceived of as the set of possible worlds that the participants in a conversation all publicly consider possible candidates for being the actual world. If a proposition holds in every world in the context set, it is PRESUPPOSED. Here is how Stalnaker characterizes it:

> Roughly speaking, the presuppositions of a speaker are the propositions whose truth he takes for granted as part of the background of the conversation. A proposition is presupposed if the speaker is disposed to act as if he assumes or believes that the proposition is true, and as if he assumes or believes that his audience assumes or believes that it is true as well. Presuppositions are what is taken by the speaker to be the COMMON GROUND of the participants in the conversation, what is treated as their COMMON KNOWLEDGE or MUTUAL KNOWLEDGE...
>
> It is PROPOSITIONS that are presupposed – functions from possible worlds into truth-values. But the more fundamental way of representing the speaker's presuppositions is not as a set of propositions, but rather as a set of possible worlds, namely those compatible with what is presupposed. This set, which I will call the CONTEXT SET, is the set of possible worlds recog-

> nized by the speaker to be the "live options" relevant to the conversation. A proposition is presupposed if and only if it is true in all of these possible worlds.
>
> The motivation for representing the speaker's presuppositions in terms of a set of possible worlds in this way is that this representation is appropriate to a description of the conversational process in terms of its essential purposes. To engage in conversation is, essentially, to distinguish among alternative possible ways that things may be. The presuppositions define the limits of the set of alternative possibilities among which speakers intend their expressions of propositions to distinguish.

When an assertion is made, and all of the interlocutors agree to it, the contents of the assertion become part of the common ground, that is, they enter the context set. But an assertion is only felicitous when its presuppositions already hold in the context set.

Let us say that the presuppositions of a sentence are SATISFIED in a given context if the context entails the presuppositions. This definition depends on a notion of entailment that can hold between contexts and sentences which we must make precise. Recall that a sentence ϕ ENTAILS another sentence ψ (written $\phi \vDash \psi$) if and only if whenever ϕ is true, ψ is true. For most of the book so far, we have adopted an extensional theory, and let sentences denote truth values. Now let us switch to an intensional view and let sentences denote propositions, i.e. sets of possible worlds. Then we can say that ϕ entails ψ if and only if the proposition expressed by ϕ is a subset of the proposition expressed by ψ: every ϕ-world is a ψ-world. For example, suppose that John is president in w_1, w_2, and w_3, so the proposition expressed by 'John is president' is:

$$\{w_1, w_2, w_3\}$$

Assume further that in every world, John is a child. Thus a child is president in all of these worlds. But there are also other worlds

where Mary, who is also a child, is the president. Call these w_4 and w_5. Then the proposition expressed by 'A child is president' is:

$$\{w_1, w_2, w_3, w_4, w_5\}$$

Since

$$\{w_1, w_2, w_3\} \subseteq \{w_1, w_2, w_3, w_4, w_5\}$$

'John is president' entails 'A child is president'. All worlds in which the former holds are worlds in which the latter holds.

Now, what does it mean for a sentence to be entailed by a context? As we said above, a context consists of all of the information that is presupposed – in other words, all of the information that is agreed upon, or taken for granted. We could think of this information as a set of sentences, or as the set of propositions expressed by these sentences. Or, as Heim (1983c, 399) puts it:

> A context is here construed more or less... as a set of propositions, or more simply, as a proposition, namely that proposition which is the conjunction of all the elements of the set.

If propositions are sets of possible worlds, then what is the conjunction of a set of propositions? Here is a concrete example:

$P = \{w_1, w_2, w_3\}$	['John is president']
$Q = \{w_1, w_2, w_3, w_4, w_5\}$	['A child is president']
$R = \{w_2, w_3\}$	['John has a girlfriend']
$S = \{w_1, w_4\}$	['Mary is sick']
$W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}\}$	[the set of all worlds]

What is the conjunction of P and S, the conjunction of the proposition that John is president and the proposition that Mary is sick? It is the set of worlds where both propositions are true. That is the intersection (not the union).

$$\begin{aligned} & P \cap S \\ = \; & \{w_1, w_2, w_3\} \cap \{w_1, w_4\} \\ = \; & \{w_1\} \end{aligned}$$

So the context will constitute a set of possible worlds, those possible worlds in which all of the presupposed facts hold, i.e., the intersection of all of the agreed-upon propositions.

On the dynamic view, the denotation of a sentence is constituted by its potential to update the context: a CONTEXT CHANGE POTENTIAL, rather than a characterization of the world. We can say that the update CRASHES when the presuppositions of the sentence are not satisfied. Let us write:

$$c + \phi$$

to denote the result of updating a context c with the proposition expressed by ϕ. Ignoring assignment functions, if we take the denotation of a sentence to be a set of possible worlds, the update that a sentence makes is to narrow down the context set to just those in which the proposition expressed by the sentence holds. A sentence like *John is happy*, for example, eliminates all worlds where John is not happy from the context set.

Exercise 1. Suppose that the context c consists of the following worlds: $\{w_1, w_2, w_3, w_4, w_5\}$ and in these worlds it is raining: $\{w_2, w_4\}$. What is the result of updating c with *It is raining*?

Suppose that we have a sentence like *John's son is bald*, which presupposes that John has a son. If there are some worlds in the context set where John does not have a son, then the presuppositions of the sentence are not satisfied in the context set. In such a situation, we say that the context set does not ADMIT the sentence. Karttunen's idea is that *in order for a context to admit a sentence, the context must entail the presuppositions of the sentence.* Admittance is defined in terms of SATISFACTION:

(4) **Satisfaction**
Let P_ϕ be the set of worlds where the presuppositions of ϕ

are satisfied. A context c satisfies the presuppositions of ϕ if $c \subseteq P_\phi$.

(5) **Admittance**
A context c ADMITS ϕ if and only if c satisfies the presuppositions of ϕ.

Now, given a context that does not satisfy the presuppositions of a given sentence, it is easy enough to repair it so that the presuppositions are taken for granted; this process is called GLOBAL ACCOMMODATION.[2] But the idea is nevertheless that the update cannot proceed until the context is such that all the presuppositions of the sentence are satisfied.

A simple, non-compound sentence will have a set of BASIC PRESUPPOSITIONS. For example, *Both Bill's children are bald* presupposes that Bill has exactly two children; this is a basic presupposition of this non-compound sentence. Non-compound sentences are admitted by a context as long as the context entails all of their basic presuppositions:

(6) **Admittance conditions for non-compound sentences**
If ϕ is a simple, non-compound sentence, then a context c admits ϕ if and only if c satisfies the basic presuppositions of ϕ. (Karttunen, 1974, 184)

Exercise 2. Assume the following:

$P = \{w_1, w_2, w_3\}$ ['John is president']
$Q = \{w_1, w_2, w_3, w_4, w_5\}$ ['A child is president']
$R = \{w_2, w_3\}$ ['John has a girlfriend']
$S = \{w_1, w_4\}$ ['Mary is sick']
$W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}\}$ [the set of all worlds]

[2] As opposed to LOCAL ACCOMMODATION, which has been posited as a last-resort mechanism that allows interpreting a presupposition locally under semantic operators when it cannot project for some reason. We will not discuss local accommodation in this book.

Suppose that the sentence ϕ = *Both Bill's children are bald* presupposes that Bill has exactly two children. Suppose that in worlds $w_1 ... w_8$, only John and Sue are children of Bill, but in w_9 and w_{10}, John, Mary, and Sue are. So the proposition that Bill has exactly two children (call it K) is the set of worlds $w_1 ... w_8$:

$$K = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$$

K is a basic presupposition of ϕ; let us pretend that it is the only one. So

$$P_\phi = K = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$$

Since our sentence ϕ is a simple, non-compound sentence, it is admitted in contexts c that entail K.

- Suppose that $c = P \cap S$. Does c admit ϕ? Why or why not?
- Suppose instead that $c = W$. Does c admit ϕ? Why or why not?

Now, consider (again) the contrast between the following two conditional sentences:

(7) If baldness is hereditary, then John's son is bald.
$\gg$ John has a son.

(8) If John has a son, then John's son is bald.
$\not\gg$ John has a son.

In the first example, the sentence as a whole presupposes that John has a son (as indicated by the symbol $\gg$). In the second example, the sentence as a whole does not at all convey that the speaker believes that John has a son. The speaker appears quite open to the possibility that he does not. Again, in a conditional sentence of the form *If A then B*, if the antecedent A satisfies the

presuppositions of *B*, then the conditional as a whole does not carry the presuppositions of *B*.

Karttunen (1974) makes sense of this by imagining that we first update the global discourse context with *A*, and that it is in this temporary, hypothetical context that the presuppositions of *B* have to be satisfied. For conditionals, Karttunen proposes the following:

(9) **Admittance conditions for a conditional sentence**
Context c admits "If ϕ then ψ" just in case (i) c admits ϕ, and (ii) $c{+}\phi$ admits ψ. Here $c{+}\phi$ designates 'c updated with ϕ'. The result of this update will be the same as if ϕ is asserted in context c; it will be defined if the presuppositions are satisfied, and if so, it will be the result of eliminating all worlds where ϕ is not true.

Consider the following examples:

(10) If Bill has exactly two children, then both his children are bald.

(11) If Bill is bald, then both his children are bald.

Assume the following:

$A = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$ ['Bill has exactly 2 children']
$W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}\}$ [the universe]

Suppose that $c = W$. Does c admit (10)? According to the admittance conditions for conditional sentences in (9), it does just in case (i) c admits *Bill has exactly two children* and (ii) $c{+}$*Bill has exactly two children* admits *both his children are bald.* Since *Bill has exactly two children* carries no presuppositions, the first condition is satisfied. What about the second condition? The result of updating c with *Bill has exactly two children* is the set *A*, the set of worlds where Bill indeed has exactly two children. Now the ques-

tion is whether this set, A, admits the non-compound sentence *both his children are bald*. Since it is a non-compound sentence, the rule for non-compound sentences (6) applies. What *both his children are bald* presupposes is that Bill has exactly two children. This is satisfied in all of the worlds in A, so the second condition is satisfied as well. Hence c does admit (10). But the same does not hold for (11).

Exercise 3. Refine c (as defined in the foregoing discussion) in such a way that it does not admit (11). To do this, specify in which of the worlds in c Bill is bald. Explain step-by-step why your refined context does not admit this sentence.

Exercise 4. Refer to Figure 9.1. Does C admit *If the king has a son, then the king's son is bald*? Why or why not? Does K admit it? Why or not? Explain using the definition of admittance, and assume that the result of updating a context with *The king has a son* is the intersection of A with the context *if the presuppositions of 'The king has a son' are satisfied*; undefined otherwise.

So now we are in a position to explain why the presupposition that Bill has exactly two children projects in a case like (11), and not in a case like (10). In order for the conditional as a whole to be admitted by a given context, both of the conditions in (9) must be met. The first condition will be met only if the presuppositions of the antecedent are already satisified in the global context. Hence *presuppositions always project from the antecedent of a conditional*. The second condition will be met either if (i) the antecedent entails the presuppositions of the consequent or (ii) the global context already entails them. If the antecedent of the conditional does not entail the presuppositions of the consequent, then the global context must already entail them. Such is

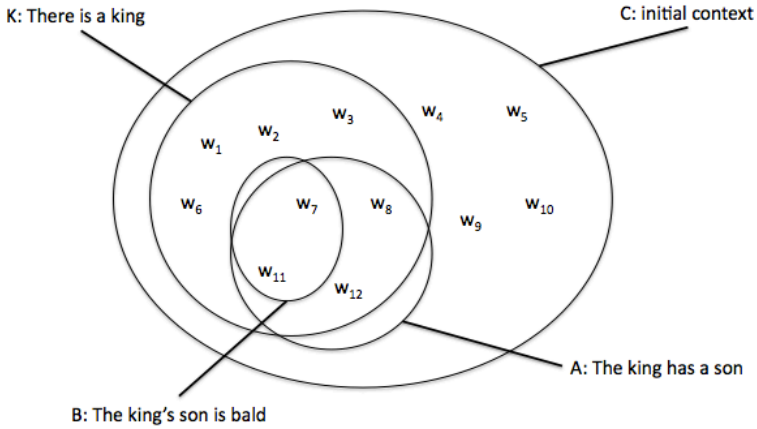


Figure 9.1: Example propositions

the situation in a case like (11), where the antecedent of the conditional does *not* entail the presuppositions of the consequent. In order for that sentence to be admitted in a given context, the context must already entail the presuppositions of the consequence. Hence the presuppositions project in that case.

Another way of putting Karttunen's insight is as follows: The global context incremented by the antecedent is the LOCAL CONTEXT for the consequent. This idea is quite general. We can identify a range of local contexts (*c* here stands for the global context):

- the consequent of a conditional → $c+$ the antecedent
- the second conjunct in a conjunction → $c+$ the first conjunct
- the second disjunct in a disjunction → $c+$ the negation of the first disjunct
- the complement of a propositional attitude verb → the beliefs of the holder of the propositional attitude (e.g. *Hans*

wants the ghost in his attic to be quiet tonight presupposes that Hans believes that there is a ghost in his attic)

In general:

(12) A context c admits a sentence S just in case each of the constituent sentences of S is admitted by the corresponding local context.(Heim, 1983c, 399)

For example, consider (25) from the previous chapter, repeated here:

(13) Either Geraldine is not a catholic or she has stopped attending services on Sundays.

Here we have a disjunction. The local context for the second disjunct is $c+$ the negation of the first disjunct. The first disjunct (*Geraldine is not a catholic*) is itself negated; let us assume that the negation of the negated sentence can be obtained simply by removing the 'not', so the local context for the second disjunct is $c+$ *Geraldine is a catholic*. Suppose it is part of the common ground in the global context that all catholics attend services on Sundays. Then the local context entails that Geraldine attends services on Sundays. The consequent, *she has stopped attending services on Sundays*, presupposes that Geraldine attends services on Sundays. Since the local context entails this proposition, the global context need not entail it, so the presupposition is filtered out.

Exercise 5. Give another example of a disjunction in which the negation of the antecedent entails the presuppositions of the consequent, and explain how the presuppositions of the consequent get filtered out.

9.3 Pronouns with indefinite antecedents

Another important motivation for dynamic semantics comes from pronouns with indefinite antecedents. In dynamic semantics, an indefinite noun phrase like *a man* introduces a new DISCOURSE REFERENT into the context, and an anaphoric pronoun or definite description picks up on the discourse referent.

One of the main motivations for dynamic semantics comes from examples involving pronouns whose antecedents are indefinite descriptions, as in the following two-sentence discourse:

(14) 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 Heim (1982b) discusses. First, it correctly captures the fact that (15) does not imply that there is a dog that John and Mary are both friends with.

(15) John is friends with a dog and Mary is friends with a dog.

If we assumed that *a dog* referred to some dog, then we would predict this sentence to have that implication. Second, Russell's analysis correctly captures the fact that (16) does not say regarding some particular dog that it came in, in contrast to (17), which has a proper name referring to a dog and does have that implication.

(16) It is not the case that a dog came in.

(17) It is not the case that Fido came in.

Third, Russell's analysis correctly captures the fact that (18) can be true even if it is not the case that there is some particular dog that everybody owns, while (19) does not have that implication.

(18) Every child owns a dog.

(19) Every child owns Fido.

If *a dog* referred to a dog then (18) should mean that every child owns that dog, as in (19).

However, there are some problems. If we analyze example (14) 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):

(20) $\exists x[\mathsf{Cat}(x) \wedge \mathsf{Found}(\mathsf{j}, x)] \wedge \mathsf{RanAway}(v_3)$

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 6. Give LF trees and derivations for the two sentences in (14). (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:

(21) $\exists x[\mathsf{Cat}(x) \wedge \mathsf{Found}(\mathsf{s}, x) \wedge \mathsf{RanAway}(x)]$

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:

(22) a. A man fell over the edge.
b. He didn't fall; he jumped.

(23) a. A dog came in.
b. What did it do next?

What would a Geachian analysis be for a case like (22)? 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. Example (23) presents a similarly puzzling challenge.

Another argument that Heim makes against the Geachian analysis is based on the following example:

(24) Susan owns some sheep. Harry vaccinated them.

This sentence should be false in a situation where Susan 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 Susan owns x and Harry vaccinated x', which would be true in such a situation. But the English sentence would not be, so this is not a welcome prediction.

Third, Geach's proposal would mean that existential quanti-

fiers have different scope properties from other quantifiers. Consider the following examples:

(25) A dog came in. It lay down under the table.

(26) Every dog came in. #It lay down under the table.

(27) No dog came in. #It lay down under the table.

In neither (26) nor (27) can *it* be bound by the quantifier in the first sentence.[3] Heim (1992, 17) concludes:

> 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:

(28) If a farmer owns a donkey, then he beats it.

[3]There is a phenomenon called *telescoping*, counterexemplifying the generalization that *every* cannot take scope beyond the sentence boundary. Examples 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")

This example is naturally interpreted as a universal statement, representable as follows:

(29) $\forall x \forall y[[\mathsf{Farmer}(x) \wedge \mathsf{Donkey}(y) \wedge \mathsf{Owns}(x,y)] \rightarrow \mathsf{Beats}(x,y)]$

But the representation that we would derive for it using the assumptions that we have built up so far would be:

(30) $[\exists x \exists y[\mathsf{Farmer}(x) \wedge \mathsf{Donkey}(y) \wedge \mathsf{Owns}(x,y)]] \rightarrow \mathsf{Beats}(x',y')$

where the existential quantifiers have scope only over the antecedent of the conditional. This analysis leaves the variables introduced by the pronouns (the x' and y') unbound; clearly it does not deliver the right denotation.

Similar problems arise with indefinite antecedents in relative clauses:

(31) Every man who owns a donkey beats it.

Exercise 7. Give a representation in L_λ capturing the intuitively correct truth conditions for (31). Then give an LF tree and a derivation for (31) 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:

(32) 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:

(33) 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.

(34) Sometimes, if a cat falls from the fifth floor, it survives.
$\Longleftrightarrow$ Some cats that fall from the fifth floor survive.

(35) 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 con-

> sists 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."[4] Thus a discourse 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:

(36) 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. Examples (26) and (27) above provide further examples 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 in-

[4] 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.

troduce aren't immortal. A good theory should account for both sides of this tension.

9.4 File change semantics

Heim's (1982b) FILE CARD SEMANTICS conceptualizes discourse referents as file cards, very much building on Karttunen's metaphor. In file card semantics, an indefinite introduces a new file card. Subsequent anaphoric reference updates the file card. For example, consider the discourse in (37):

(37) 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:

(38)

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 that the descriptive content (*dog*) serves merely to identify the appropriate discourse referent.

Exercise 8. Add a sentence to (37) 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 (38) 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 and Sue are women, 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 (38). 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.

9.5 Discourse Representation Theory

File card semantics is not the only dynamic theory of meaning; another very well-developed and well-known one 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 (37) would look as follows:

x y z
Woman(x)
Dog(y)
Paddle(z)
Bit(y,x)
Hit-with(x,y,z)
Ran-away(y)

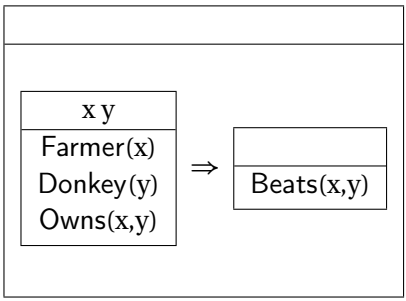
Just as in file card semantics, this kind of structure is thought to be built up over the course of a discourse, and the denotation of a sentence can be seen as its potential to affect any DRS representing the current state of the discourse. A DRS has two parts:

- 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:

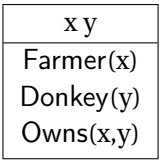
x y
Farmer(x)
Donkey(y)
Owns(x,y)

The same sentence used as the antecedent of a conditional would appear as a DRS contained in a larger DRS, as follows:



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.

Truth in DRT is defined relative to a DRS. A DRS is defined to be TRUE relative to a model if there is an embedding that VERIFIES it in the model. Which embeddings verify a given DRS is determined by semantic clauses for DRSs. But to give an idea, consider the following DRS:



A function g verifies this DRS with respect to model M if:

- the domain of g contains at least x and y
- according to M it is the case that $g(x)$ is a farmer, $g(y)$ is a donkey, and $g(x)$ owns $g(y)$.

As in predicate logic, we have models $M = \langle D, I \rangle$. I assigns an extension to every predicate (Farmer, Donkey, Owns, etc.). I(Farmer)

will be a set of individuals; $I(\text{Owns})$ will correspond to a relation. So g verifies $\text{Farmer}(x)$ with respect to model $M = \langle D, I \rangle$ if and only if $g(x) \in I(\text{Farmer})$. What this means is that an embedding g verifies the DRS for *A farmer owns a donkey* if it assigns x to a farmer, and y to a donkey that the farmer owns.

In general, verification of a DRS is defined as follows:

(39) **Verification of a DRS**
Embedding g VERIFIES DRS K in model M if and only if g verifies every condition in K, and the domain of g includes every discourse referent in the universe of K.

Whether or not a given embedding g verifies a given condition depends on the nature of the condition. Let us use the notation

$$M, g \vDash \phi$$

to denote 'g verifies condition ϕ in model M'. The rule for deciding whether a given embedding verifies a condition like Farmer(x), where a predicate applies to an argument, is defined as follows:

(40) **Verification of a predication condition**
$M, g \vDash \pi(x)$ iff $g(x) \in I(\pi)$ where π is a predicate and model $M = \langle D, I \rangle$.

Recall that an indefinite will introduce a new discourse referent into the discourse, and add the condition that the descriptive content apply to the discourse referent, so *A farmer owns a donkey* will be represented:

x y
Farmer(x)
Donkey(y)
Owns(x,y)

According to the rules that we have set out, this DRS will be true in M if there is an embedding g with a domain that includes x and y,

which verifies all three of the conditions, in other words, if there are indeed x and y such that x is a farmer and y is a donkey and x owns y.

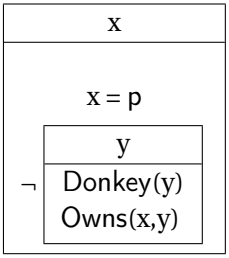
Exercise 9. Under this treatment, indefinites in unembedded sentences like *A farmer owns a donkey* are interpreted essentially as existential quantifiers. Suppose that your friend doesn't understand why this is so, and explain it to them so that they say 'Aha!'.

Another kind of atomic condition is equality:

(41) **Verification of an equality condition**
$M, g \vDash x = y$ iff $g(x) = g(y)$

This says that embedding g verifies the condition 'x = y' in model M if $g(x)$ is the same entity as $g(y)$.

Verifying a negated condition such as the following is a bit more complex. Suppose that this is the representation for *Paul does not own a donkey*.



Intuitively, this should be true if and only if there is no way to assign a value to x such that x is Paul, and there is some individual y such that y is a donkey and x owns y. This is defined with the help of some auxiliary notions:

- **Compatibility**
  We say that two functions f and g are COMPATIBLE if they

assign the same values to those arguments for which they are both defined. I.e., f and g are compatible if for any a which belongs to the domain of both f and g:

$$f(a) = g(a)$$

- **Extension**
  g is called an EXTENSION of f if g is compatible with f and the domain of g includes the domain of f.

Thus if g is an extension of f then f and g assign the same values to all arguments for which f is defined, while g may (though it need not) be defined for some additional arguments as well.

Returning to negation:

(42) **Verification of a negated condition**
An embedding function f verifies a condition of the form $\neg K$ with respect to model M iff there is no function g such that:

•g extends f

•g verifies K

Thus, for example, a function f verifies the negated condition in the DRS for *Paul does not own a donkey* iff:

- f verifies x = p, and
- There is no function g such that: (i) g extends f, and (ii) g verifies

y
Donkey(y)
Owns(x,y)

This gives us results for negated sentences containing indefinites on par with Russell's treatment: Just as with negated existentials, a negated sentence containing an indefinite that takes scope under the negation will be true only if there is no object in the

model satisfying the relevant description. Furthermore, the fact that the discourse referent is introduced in a DRS that is nested within another DRS, and, as it were, "shielded" from the top level by a negation symbol, gives us the tools to account for the fact that *a donkey* does not license an antecedent in a later sentence. We will not go through how this works here; suffice it to say that the discourse referent is not ACCESSIBLE for subsequent anaphora in this position.

Exercise 10. Partially specify a model $M = \langle D, I \rangle$ where *Paul does not own a donkey* is true, by specifying the value of I for the relevant constants. Then give an embedding function f that verifies the negated condition in the DRS for *Paul does not own a donkey* in M, and explain why it verifies that condition.

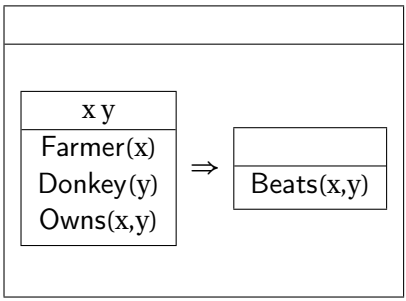
The semantics of conditionals uses the concept of extensions among embedding functions as well.

(43) **Verification of a conditional condition**
An embedding function f verifies a condition of the form $K \Rightarrow K'$ with respect to model M if and only if: For all extensions g of f that verify K, there is an extension h of g that verifies K'.

The intuitive idea is something like the following: To verify a conditional statement, first consider what kind of embedding would be necessary to verify the antecedent. Now consider whether or not the consequent has to hold, given that embedding.

It turns out that this semantics for conditionals allow for a unified account of indefinites across the full range of uses we have seen. In particular, although unembedded indefinites get an existential interpretation, indefinites acquire universal import in conditionals, and indefinites can bind from antecedent to consequent. Consider the DRS for the donkey sentence:

(44)



For an arbitrary embedding f, we want to determine whether every extension g of f that verifies the antecedent DRS has an extension h of g that verifies the consequent DRS. Suppose we have a model $M = \langle D, I \rangle$ in which the following is the case:

(45) $I(\text{paul}) = a$
$I(\text{Farmer}) = \{a, b, c\}$
$I(\text{Donkey}) = \{d, e, f\}$
$I(\text{Owns}) = \{\langle a, d\rangle, \langle b, e\rangle, \langle b, f\rangle\}$
$I(\text{Beats}) = \{\langle a, d\rangle, \langle b, e\rangle, \langle b, f\rangle\}$

Let f be the null embedding, which has the empty set as its domain. The extensions g of f that verify the antecedent are the ones that assign x to a farmer and y to a donkey that is owned by the farmer. For example, this criterion would be satisfied by an embedding that assigns x to a and y to d, like this:

$$g = \begin{bmatrix} x & \rightarrow & a \\ y & \rightarrow & d \end{bmatrix}$$

Now, in this case, there is an extension h of g that verifies the consequent, namely g itself, since a beats d. In general, since the Owns relation is exactly the same as the Beats relation, given an assignment g that verifies the antecedent, there will always be an extension h of g that verifies the consequent, namely g itself. In other words, for every given case where we have a pair x and y where x is a farmer and y is a donkey owned by the farmer, the farmer in that pair also beats that donkey. If that condition did

not hold, then the condition would be false. Hence we have *universal import* for indefinites in conditional sentences.

Exercise 11. Let g be the empty embedding $\varnothing$. Using the assumptions about the model given in (45), list all of the embeddings g that verify the antecedent DRS in (44). For each of those embeddings, give an embedding h that verifies the consequent.

Exercise 12. Change the model specified in (45) so that the condition in (44) is not satisfied, and name the embedding g that verifies the antecedent that does not have an extension h that verifies the consequent.

Exercise 13. Draw a DRS for *If a farmer beats a donkey, then he beats a friend of the donkey,* and give a model in which the conditional is (non-trivially) satisfied. Give an example of an embedding g and an extension h of g such that g verifies the antecedent and h verifies the consequent.

9.6 Compositional DRT

We have not yet touched on how composition works, i.e., how files or DRSs are to be constructed from an LF representation. Both file change semantics and DRT look quite different from the systems we have presented in previous chapters. 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 for-

malization 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.

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.[5] 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 a 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, r', r''.

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 dis-

[5]Muskens (1996) uses the terms *unspecific discourse referents* for our discourse referents and *specific discourse referents* for our names.

course referents cannot. [6]

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*. For reasons that will become clear shortly, we will use i, j, j', j'', etc. and o as variables over assignments.

We assume that every assignment maps every name to the relevant individual in the domain, so that for any assignment i we have i^{j}_r =John, i^{m}_r =Mary etc. 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:

(46) **Definition**
Let i and o be two assignments and r a discourse refer-

[6] 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.

ent. We write $i[r]o$ 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 9.4, 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 Schönfinkeled 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 (37), 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 treat pronouns as essentially denoting discourse referents; is not crucial, and we could, instead let pronouns denote *variables* over discourse referents (Muskens, 2011).

(47) a. A^{r} dog bit a woman$^{r'}$.
b. She$_{r'}$ hit him$_{r}$ with a$^{r''}$ paddle.
c. It$_{r''}$ broke in half.
d. The$_{r}$ dog ran away.

The context-change potential of sentence (47a) consists in introducing two discourse referents r and r', and updating the context such that whichever entity r refers to is a dog, and whichever entity r' refers to is a woman that was bitten by r. 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.

Formally, sentence (47a) denotes the following context-change potential:

(48) $\lambda i \lambda o . \exists j . i[r]j \wedge j[r']o$
$\wedge \mathsf{Dog}(o(r)) \wedge \mathsf{Woman}(o(r')) \wedge \mathsf{Bite}(o(r), o(r'))$

In words, this is (the Schönfinkeled 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 and r', and furthermore o maps r to some dog and r' to some woman whom that dog bit.

To keep things readable, from this point onwards for any assignment j and discourse referent r, we will write the lookup operation $j(r)$ as j_r. Thus the relation above can be written as follows:

(49) $\lambda i \lambda o . \exists j . i[r]j \wedge j[r']o$
$\wedge \mathsf{Dog}(o_r) \wedge \mathsf{Woman}(o_{r'}) \wedge \mathsf{Bite}(o_r, o_{r'})$

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 9.4, in which Pug bit Joan, the following pair of assignments i^1 and o^1 will stand in the relation (49):

$$i^1 = \begin{bmatrix} r & \rightarrow & \mathsf{Bill} \\ r' & \rightarrow & \mathsf{Bill} \\ r'' & \rightarrow & \mathsf{Mary} \end{bmatrix} \qquad o^1 = \begin{bmatrix} r & \rightarrow & \mathsf{Pug} \\ r' & \rightarrow & \mathsf{Joan} \\ r'' & \rightarrow & \mathsf{Mary} \end{bmatrix}$$

The values that i^1 assigns to r and r' are irrelevant, and so is the value that both i^1 and o^1 assign to r''. These values have been filled in only for concreteness. Many other assignments than i^1 and o^1 stand in the relation denoted by (49). For example, since the values that the input assignment assigns to r and r' are irrelevant, i^1 and o^1 could be replaced by any other pair of assignments, so long as they map r'' to the same value as each other and the second assignment still maps r and r' to the same values as o^1 does. This means that the relation (49) 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 (49) succeeds on every input assignment.

The next sentence, (47b), denotes the following context-change potential:

(50) $\lambda i \lambda o.\, i[r'']o \wedge \text{Hit-with}(o_{r'}, o_r, o_{r''}) \wedge \text{Paddle}(o_{r''})$

This relation holds between assignments i and o just in case they differ at most in what they assign to r'', and furthermore o maps r'' to some paddle which was used by whatever o assigns to r' in order to hit whatever o assigns to r.

What kinds of assignments stand in this relation? Since o and i must agree in everything except possibly r'', they must both assign the same value to r, and likewise for r'. As for r'', 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 (50), i^2 must be exactly as below except that r'' could also be mapped to any other value than Mary; and o^2 must be exactly as given below.

$$i^2 = \begin{bmatrix} r & \rightarrow & \text{Pug} \\ r' & \rightarrow & \text{Joan} \\ r'' & \rightarrow & \text{Mary} \end{bmatrix} \qquad o^2 = \begin{bmatrix} r & \rightarrow & \text{Pug} \\ r' & \rightarrow & \text{Joan} \\ r'' & \rightarrow & \text{Paddle} \end{bmatrix}$$

Because of the constraints it imposes, the relation (50) 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 $She_{r'}$ and him_r in (47b)).

Typically, the previous discourse will supply input assignments on which such sentences succeed. For example, the pronouns in (47b) have their antecedents in the previous sentence (47a).

To connect pronouns with their antecedents, we now combine the two denotations (49) and (50) by an operator called *sequencing* and written as a semicolon (;). This operator is introduced here as a shorthand:

(51) $; =_{def} \lambda p \lambda q \lambda i \lambda o.\, \exists j.\, p(i)(j) \wedge q(j)(o)$

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 (49) with (50). That relation is the following:

(52) $\lambda i \lambda o \exists j \exists j' . i[r]j \wedge j[r']j'$
$\wedge \mathsf{Dog}(j'_r) \wedge \mathsf{Woman}(j'_{r'}) \wedge \mathsf{Bite}(j'_r, j'_{r'})$
$\wedge j'[r'']o \wedge \mathsf{Hit\text{-}with}(o_{r'}, o_r, o_{r''}) \wedge \mathsf{Paddle}(o_{r''})$

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, r', and r'' 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:

(53) **Bridging Principle 1**
Let i be an assignment and ϕ be a term of type **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:

(54) **Bridging Principle 2**
Let ϕ be a term of type **t** without unresolved anaphoric dependencies. ϕ is true iff it is true relative to every input assignment (in the sense of Bridging Principle 1); otherwise 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 (52) is true just in case for every assignment i there is an assignment o that is just like i except that it maps r, r', and r'' 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 to the dog, r' to the woman, and r'' 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 (55a) would have the same truth conditions as Heraclitus' famous aphorism in(55b).

(55) a. It$_r$ is in flux.
b. Everything is in flux.

This is because (55a) is true relative to any input assignment that maps r to something in flux. Suppose now that everything is in flux; then, and only then, every assignment whatsoever will map r to something in flux. Suppose instead that some things are in flux and others aren't; in that case, some assignments will map r to something in flux, while others will not. Accordingly, (55a) 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:

(56) 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:

(57) $\mathit{John} \rightsquigarrow \mathsf{john}$

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.

(58) $a^r \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j\,.\, i[r]j \wedge (R(j_r); N(j_r))(j)(o)$

For the sake of readability, from here on we will write $\phi(i)(o)$ as $i[\phi]o$, for any formula ϕ of type $\mathbf{t}$; thus this above simplifies as follows:

(59) $a^r \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j\,.\, i[r]j \wedge j[R(j_r); N(j_r)]o$

We can also spell out the sequencing shorthand to make things clearer:

(60) $a^r \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j\,.$
$i[r]j \wedge \exists j'\,.\, j[R(j_r)]j' \wedge j'[N(j_r)]o$

Using this entry and two instances of function application, sentence (61a) evaluates to (61b), which is equivalent to (61c) due to the equivalences between assignments:

(61) a. A^r dog ran away.
b. $\lambda i \lambda o\,.\, i[r]j \wedge \exists j'\,.\, \mathsf{Dog}(j_r) \wedge j = j'$
$\wedge\, \mathsf{Run\text{-}away}(j_r) \wedge j' = o$
c. $\lambda i \lambda o\,.\, i[r]o \wedge \mathsf{Dog}(o_r) \wedge \mathsf{Run\text{-}away}(o_r)$

That (61b) is so much more complicated than (61c) is due to the fact that neither the restrictor nor the nuclear scope of the indefinite *a* in (61a) 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 (59). 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:

(62) A^{r} $[_{Restr}$ farmer who had $\text{a}^{r'}$ donkey$]$ $[_{Nucl}$ beat $\text{it}_{r'}]$.

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. 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.

(63) $he_r/him_r/she_r/her_r/it_r \rightsquigarrow \lambda P \lambda i \lambda o\,.\, i = o \wedge i[P(i_r)]o$

For example, (64a) denotes (64b):

(64) a. It_r 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 (65a) is equivalent to (65b):

(65) 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, def-

inite determiners combine with a restrictor and a nuclear scope, which are both applied to the entity they refer to.

(66) $the_r \rightsquigarrow \lambda R \lambda N \lambda i \lambda o \exists j \,.\, i[R(j_r)]j \wedge j[N(j_d)]o$

Consider now a transitive verb such as *bite*. Following the same reasoning as before, we arrive at the following lexical entry:

(67) 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 ditransistive 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 $\langle\langle e, \mathbf{t}\rangle, \mathbf{t}\rangle$.

(68) 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:

(69) Preliminary entry
$hit\text{-}with \rightsquigarrow \lambda y \lambda z \lambda x \lambda i \lambda o \,.\, i = o \wedge \mathsf{Hit\text{-}with}(x, y, z)))$

(70) Final entry
$hit\text{-}with \rightsquigarrow \lambda Q' \lambda Q \lambda x \,.$
$Q'(\lambda y \lambda i \lambda o \,.\, i = o \wedge Q(\lambda z \lambda i \lambda o \,.\, i = o \wedge \mathsf{Hit\text{-}with}(x, y, z)))$

Using these entries, we can generate context change potentials for the sentences in (47). We have already seen the context-change potentials for (47a) and (47b) in (49) and (50). The one for (47c) is analogous to the one in (64b), and the one for (47d) is similar.

Exercise 14. Using the appropriate CDRT lexical entries, give a compositional derivation of the context change potential of Sentence (47d). Show the details of the derivation. Use equivalences between assignments to simplify the result as much as possible in the same manner shown in (61b) and (61c).

Exercise 15. We have seen how the context change potentials of sentences (47a) and (47b) 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 (47a) serving as input to (47b). Do the same for the transitions from (47b) to (47c), and from (47c) to (47d). 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:

(71) $not \rightsquigarrow \lambda p \lambda i \lambda o \,.\, i = o \wedge \neg \exists j \,.\, i[p]j$

Exercise 16. 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:

(72) Paul does not own a^r donkey. #It$_r$ 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:

(73) *own a^r donkey*
$\rightsquigarrow \lambda x \lambda i \lambda o . \exists j . i[r]j \wedge \mathsf{Donkey}(j_r) \wedge \mathsf{Owns}(x, j_r)$

After combining with the subject, we get:

(74) *Paul owns a^r donkey*
$\rightsquigarrow \lambda i \lambda o . . i[r]o \wedge \mathsf{Donkey}(o_r) \wedge \mathsf{Owns}(\mathsf{paul}, o_r)$

This context-change potential relates any two assignments i and o just in case they differ at most in r and r', in such a way that o maps r to a donkey that Paul owns.

The bridging principles in (53) and (54) 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:

(75) *not(Paul owns a^r donkey)*
$\rightsquigarrow \lambda i \lambda o . i = o \wedge \neg \exists j . i[r]j \wedge \mathsf{Donkey}(j_r) \wedge \mathsf{Owns}(\mathsf{paul}, j_r)$

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*

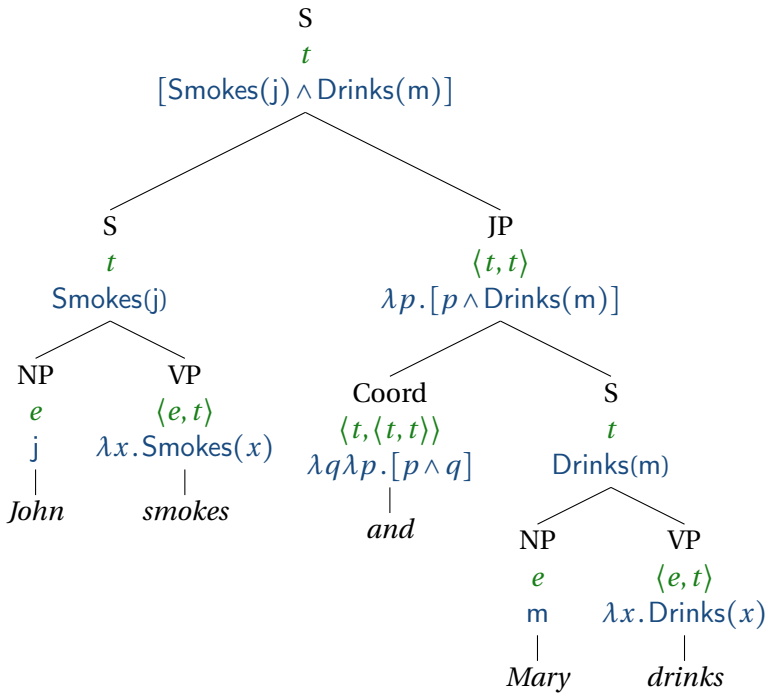
To translate these into the lambda calculus, we can simply write the following (here, p and q are variables over truth values):

(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 arrived. (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 sen-

tences. For example, VP coordination was analyzed as involving deletion of the subject of the second sentence is silent (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	$\rightarrow$	X CoordP
CoordP	$\rightarrow$	Coord X

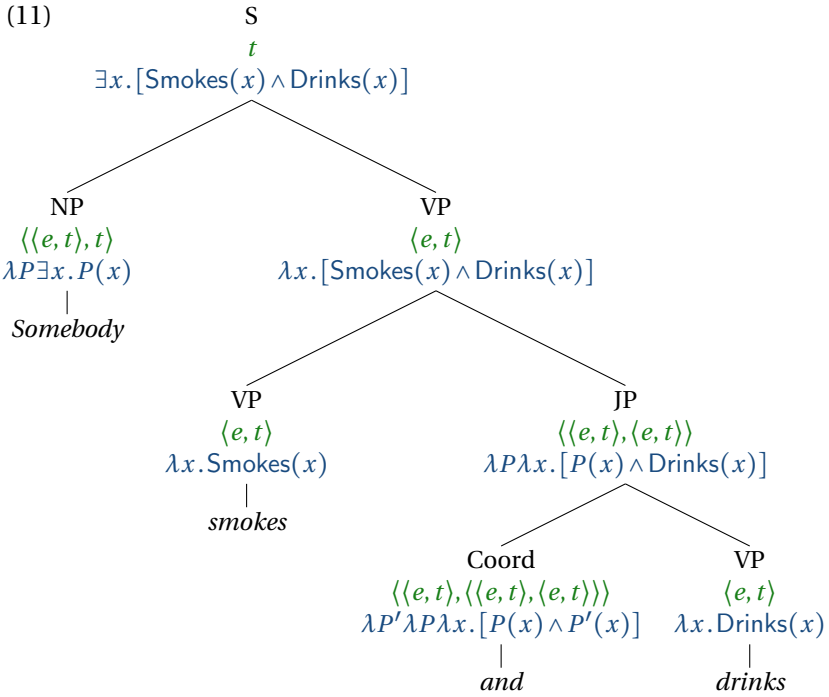
where $X \in \{\text{S, VP, DP, V,}\ldots\}$

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 *Mary loves and hates John*? Assuming that transitive verbs translate to expres-

sions of type $\langle e, \langle e, t\rangle\rangle$, that is, (Schönfinkelized) binary relations, the version of *and* that should be used to coordinate them should take two binary relations and return a new binary relation. The following entries will do that trick.

(12) a. $and_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . [R(y)(x) \wedge R'(y)(x)]$
b. $or_V \rightsquigarrow \lambda R' \lambda R \lambda y \lambda x . [R(y)(x) \vee R'(y)(x)]$

Given *loves* and *hates*, these lexical entries will produce a new relation, 'loves and hates'.

Exercise 2. Using the lexical entry for *and* above, draw the tree for *Susan caught and ate the fish*.

Noun phrase coordination (that is, coordination of DPs) can be approached in the same way. Let us first look at conjunctions of quantifiers:

(13) a. Every man and every woman arrived.
$\forall x . [\mathsf{Man}(x) \rightarrow \mathsf{Arrived}(x)] \wedge$
$\forall x . [\mathsf{Woman}(x) \rightarrow \mathsf{Arrived}(x)]$
b. A man or a woman arrived.
$\exists x . [\mathsf{Man}(x) \wedge \mathsf{Arrived}(x)] \vee$
$\exists x . [\mathsf{Woman}(x) \wedge \mathsf{Arrived}(x)]$

Since quantifiers have a higher type, they take verb phrases as arguments. This makes the entries for *and* and *or* very similar to their VP-coordinating counterparts:

(14) a. $and_{DP} \rightsquigarrow \lambda Q' \lambda Q \lambda P . [Q(P) \wedge Q'(P)]$
b. $or_{DP} \rightsquigarrow \lambda Q' \lambda Q \lambda P . [Q(P) \vee Q'(P)]$

Exercise 3. Using the lexical entries above, draw the trees for *Every man and every woman arrived* and *A man or a woman arrived*.

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 arrived.

The translation we should obtain for this sentence is as follows:

$$[\mathsf{Arrived}(\mathsf{j}) \wedge [\forall x.\mathsf{Woman}(x) \rightarrow \mathsf{Arrived}(x)]]$$

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 be translated as follows:

$$\lambda P.P(\alpha')$$

as well.

This rule, which goes back to Montague (1973b), 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 $\langle\langle e,t\rangle,t\rangle$. In a sentence like *John arrived*, it can take the verb phrase as an argument. In a sentence like *John and every woman arrived*, we are able to conjoin it with *every woman* using the entry and_{DP}.

Exercise 4. Draw the tree for *John and every woman arrived* and derive a semantic interpretation for it compositionally.

Exercise 5. Draw a tree for *John and Mary smoke* and give a derivation that results in:

$$[\text{Smoke}(j) \wedge \text{Smoke}(m)]$$

You will need to apply the type shifter once on each conjunct.

10.2 Mereology

All of the occurrences of *and* that we have seen so far can be related to the denotation of logical conjunction. The schema in **??** encapsulates this relation by reducing various uses of *and* to logical conjunction. This will not work in every case, though. Consider the following example.

(16) John and Mary are a happy couple.

There is no obvious way to formulate the truth conditions of (16) using logical conjunction. It cannot be represented as:

$$[\text{Happy-couple}(j) \wedge \text{Happy-couple}(m)]$$

since this would entail $\text{Happy-couple}(j)$ as well as $\text{Happy-couple}(m)$. 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 do you appreciate most with your friends?" was... "that they are very 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 Mary* 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-Mary (where in set theory, we would use $\in$), and also between John-and-Mary and John-and-Mary-and-Sue (where in set theory, we would use $\subseteq$).

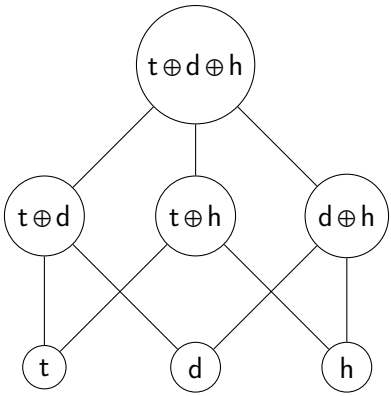
Mereology also provides an operator, $\oplus$, that allows us to put individuals together to form collections. The formal objects that represent these collections in mereology are called SUMS. For example, the collection John-and-Mary is represented formally as the sum $\mathsf{j} \oplus \mathsf{m}$. Collective predicates apply directly to such sums. For example, *John and Mary are a happy couple* can be represented as $\mathsf{Happy\text{-}couple}(\mathsf{j} \oplus \mathsf{m})$. Since the sum $\mathsf{j} \oplus \mathsf{m}$ is of type e, the type of the VP is $\langle e, t \rangle$ as usual.

Exercise 6. 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 Mary met*.

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, $\oplus$) defined on it. Figure 10.1 illustrates such a structure. The circles stand for the individuals Tom, Dick, and Harry, 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 Tom, Dick, and Harry, as ATOMIC INDIVIDUALS; the other circles stand for individuals which are not atomic. 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. (This 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.) Transitivity means that if a is part of b and b is part of c, then a is also part of c. For example, according to Figure 10.1, t is part of $\mathsf{t} \oplus \mathsf{d}$, and $\mathsf{t} \oplus \mathsf{d}$ is part of $\mathsf{t} \oplus \mathsf{d} \oplus \mathsf{h}$; therefore, by transitivity, t is also part of $\mathsf{t} \oplus \mathsf{d} \oplus \mathsf{h}$. Finally, antisymmetry means that two distinct things cannot both be part

of each other. This condition is very intuitive. For example, since t is part of $t \oplus d$, it follows that $t \oplus d$ is not part of t.

Figure 10.1: An algebraic structure



The branch of formal semantics that uses algebraic structures and parthood relations to model various phenomena is known as algebraic semantics. The fundamental assumption in algebraic semantics is that any nonempty set of things of the same sort (for example individuals or events) has one and exactly one sum. So far, we have only considered one sort, namely individuals (type *e*). We will assume that all individuals, including sums, will be of type *e*. To express that the atomic individual Tom is part of the sum individual Tom-and-Dick, we will write $t \leq t \oplus d$. In structures like the one depicted in Figure 10.1, the sum of any nonempty set of individuals *P* is always the lowest individual that sits above every element of *P* other than itself. (As we will see later, this corresponds to the mathematical notion of "least upper bound".) For example, if *P* consists of the two atomic individuals t and d, then the lowest individuals that sits above these two is $t \oplus d$. Sometimes the sum of *P* can be a member of *P*. For example, if *P* consists of t and $t \oplus d$, then its sum is $t \oplus d$ again. And if *P* consists of just one

individual, such as h, then its sum is that individual itself.

10.3 The plural

10.3.1 Algebraic closure

Coordinations of proper nouns are not the only way to talk about sums:

(17) a. Tom, Dick and Harry met.
b. Some boys met.
c. Three boys met.
d. The boys met.

In each of these three sentences, the collective predicate *met* applies to a sum x. Only (17a) fully specifies the parts of that sum, while (17b) through (17d) describe it partially. That is, we know that they are all boys, but we don't know who they are.

Just like the predicate *met*, the plural noun *boys* can be seen as denoting a predicate that applies to the sum x. What is the denotation of the noun *boys*? One way to describe it is in terms of the conditions it imposes on x, namely, *boys* requires it to be the sum of some boys. 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. This operation is captured by the notion of algebraic closure, which has been proposed to underlie the denotation of the plural (Link, 1983b):

(18) **Definition: Algebraic closure**
The algebraic closure $^*P$ of a set P is the set that contains any sum of any nonempty subset of P.

The most straightforward way to implement this idea is to identify the denotation of the plural morpheme with the "star operator":

(19) $\textit{-s} \rightsquigarrow \lambda P . {}^{*}P$

For example, suppose that we are in a model with just three boys, Tom, Dick and Harry. Then the denotation of the noun *boy* might be modeled as $\{\mathsf{t}, \mathsf{d}, \mathsf{h}\}$. The denotation of the noun *boys* is the algebraic closure of that set: $\{\mathsf{t}, \mathsf{d}, \mathsf{h}, \mathsf{t} \oplus \mathsf{d}, \mathsf{t} \oplus \mathsf{h}, \mathsf{d} \oplus \mathsf{h}, \mathsf{t} \oplus \mathsf{d} \oplus \mathsf{h}\}$. This set contains everything that is either a boy or a sum of two or more boys. It might seem strange to include individual boys in this set. After all, it sounds strange to say *Tom are boys*, 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 *boys* essentially synonymous with *two or more boys*. But *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 (19) as the denotation of the plural, and rule out *Tom are boys* on pragmatic grounds. That is, *boys* literally means *one or more boys*, and the singular is a special case of the plural. *Boy* and *boys* are in competition, and the singular form blocks the plural form because it is more specific (Sauerland et al., 2005; Spector, 2007).

Link gave plural individuals the status of first-class citizens in the logical representation of natural language. This allowed him to represent collective predicates like *meet* as predicates that apply directly to sum individuals:

(20) a. Tom (and) Dick and Harry met. $\rightsquigarrow \mathsf{Meet}(\mathsf{t} \oplus \mathsf{d} \oplus \mathsf{h})$
b. Some boys met. $\rightsquigarrow \exists x . [{}^{*}\mathsf{Boy}(x) \wedge \mathsf{Meet}(x)]$

As seen in (20a), 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:

(21) a. Tom is a boy and Dick is a boy. $\rightsquigarrow [\mathsf{Boy}(\mathsf{t}) \wedge \mathsf{Boy}(\mathsf{d})]$

b. Tom and Dick are boys. $\rightsquigarrow {}^{*}\mathsf{Boy}(\mathsf{t} \oplus \mathsf{d})$

Exercise 7. Draw trees for the sentences in (20) and (21b), 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 (20) and (21b).

10.3.2 Plural definite descriptions

Now, supposing that *boys* denotes the set of boy-pluralities, what does *the boys* denote? If we translate *the boys* as:

$$\iota x.\, {}^{*}\mathsf{Boy}(x)$$

then we will have a presupposition failure as long as there is more than one boy, because more than one individual will satisfy the predicate ${}^{*}\mathsf{Boy}(x)$.[1] How shall we remedy this problem?

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 sum operator is usually written with a σ (the Greek letter 'sigma'), following Link (1987). It is defined as follows:

(22) $\sigma x.\, P(x)$ is defined as:
$\iota x.\, {}^{*}P(x) \wedge \forall y[{}^{*}P(y) \rightarrow y \leq x]$

For example, $\sigma x.\, \mathsf{boy}(x)$ denotes the sum of all of the boys.

The plural definite article can then be treated as denoting this sum operator:

(23) $\text{the}_{\text{SUM}} \rightsquigarrow \lambda P.\, \sigma x.\, P(x)$

[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 *boys*, this yields:

(24) the_{SUM} boys $\rightsquigarrow \sigma x.{}^{*}\text{Boy}(x)$

In a model where the boys are Tom, Dick and Harry, this is equivalent to $\{\text{t}, \text{d}, \text{h}, \text{t} \oplus \text{d}, \text{t} \oplus \text{h}, \text{d} \oplus \text{h}, \text{t} \oplus \text{d} \oplus \text{h}\}$. As can be checked with Figure 10.1, the sum of this set, and therefore the denotation of *the boys*, is just $\text{t} \oplus \text{d} \oplus \text{h}$. This is exactly what we want.

But in other cases, such as *the boy* and *the two boys*, we run into a problem. To see this, let us first establish some assumptions about how phrases like *two boys* are interpreted. Suppose that *two* denotes the property of being a sum of exactly two atomic individuals (for which we will write $\text{Card}(x) = 2$), and that it combines with *boys* via Predicate Modification:

(25) two $\rightsquigarrow \lambda x.\text{Card}(x) = 2$

Then *two boys* will translate as follows:

(26) two boys $\rightsquigarrow \lambda x.[\text{Card}(x) = 2 \wedge {}^{*}\text{Boy}(x)]$

In our model, the set denoted by *two boys* is $\{\text{t} \oplus \text{d}, \text{t} \oplus \text{h}, \text{d} \oplus \text{h}\}$.

Exercise 8. In order to deal with sentences like *Two boys 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 boys met* using this assumption. Don't forget to include the silent determiner in the tree diagram.

Now, what about *the two boys*? If we use $\textit{the}_{\text{SUM}}$ from above, then we will get the sum of the two-boy pluralities. As a glance at

Figure 10.1 will confirm, this sum is $\mathsf{t} \oplus \mathsf{d} \oplus \mathsf{h}$. So we end up with the rather odd prediction that *the two boys* refers to this sum!

Exercise 9. Translate *The*$_{\text{SUM}}$ *boys met* and *The*$_{\text{SUM}}$ *two boys met.*

Exercise 10. In the model where the boys are Tom, Dick, and Harry, what (if anything) do the expressions *the*$_{\text{SUM}}$ *boy*, *the*$_{\text{SUM}}$ *boys*, *the*$_{\text{SUM}}$ *two boys* and *the*$_{\text{SUM}}$ *three boys* 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 boys* should give rise to a presupposition failure, because there are three boys 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 (Montague, 1979):

(27) $\quad \text{the}_{\text{SUPR}} \rightsquigarrow \lambda P \iota x[P(x) \wedge \forall y[P(y) \rightarrow y \leq x]]$

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 other element of P. This is the same procedure as the one for determining the sum of P except that, unlike the supremum, the sum of P is not required to be an element of P.

It turns out that this representation even works for the singular definite article. In any model where there is exactly one boy, the set denoted by *boy* is a singleton, and since everything is part of

itself, the representation in (27) picks out the only member of that singleton. In all other models, the ι operator will not be defined.

Exercise 11. Translate *The*$_{\text{SUPR}}$ *boys met* and *The*$_{\text{SUPR}}$ *two boys met*. This exercise can be solved in the Lambda Calculator.

Exercise 12. In the model where the boys are Tom, Dick, and Harry, what (if anything) do the expressions *the*$_{\text{SUPR}}$ *boy*, *the*$_{\text{SUPR}}$ *boys*, *the*$_{\text{SUPR}}$ *two boys* and *the*$_{\text{SUPR}}$ *three boys* 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 three kinds of predicates that apply to sums: plural nouns like *boys*, collective predicates like *met*, pluralized distributive predicates like *arrived-∅*. All these are one-place predicates. Sums can also be related by two-place predicates, as in the following sentences:

(28) 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. Tom, Dick and Harry (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 (28a) 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 (28b) (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 (28c) is true in a scenario where Tom, Dick and Harry 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 Tom owns gadget g_1, Dick owns gadget g_2, and Harry owns gadget g_3 and also gadget g_4, then the sum of Tom, Dick and Harry 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). Here we rely on the notion of a TUPLE, defined in Chapter 2 as a finite sequence of elements.

(29) **Definition: Sum of a set of tuples**
The sum of a set of tuples is the tuple whose first element is the sum of the first elements of these tuples, whose second element is the sum of the second elements of these tuples, and so on.

(30) **Definition: Tuple of an n-place predicate**
For a given n-place relation R, a tuple of R is any n-tuple $\langle x_1, x_2, \ldots, x_n, \rangle$ such that $R(x_1)(x_2)\ldots(x_n)$.

(31) **Definition: Algebraic closure of an n-place predicate**
The algebraic closure ${}^*R$ of an n-place predicate R is the set that contains any sum of any nonempty subset of tuples of R. We write ${}^*R(a,b)$ for ${}^*R(\langle a,b\rangle)$.

We can then represent cumulative readings by using the algebraic closure of transitive verbs:

(32) Tom, Dick and Harry own four toothbrushes. $\rightsquigarrow$
$\exists x_1 . {}^*\mathsf{Gadget}(x) \wedge \mathsf{Card}(x) = 4 \wedge$
${}^*\mathsf{Own}(\mathsf{t} \oplus \mathsf{d} \oplus \mathsf{h}, x)$

An example model which verifies formula 38 is the one described above, where Tom owns gadget 1, Dick owns gadget 2, and Harry owns toothbrushes 3 and 4. The n-tuples of the relation denoted by "own" are the pairs (2-tuples) $\langle t, g_1\rangle$, $\langle d, g_2\rangle$, $\langle h, g_3\rangle$ and $\langle h, g_4\rangle$. The sum of these four pairs is $\langle t \oplus d \oplus h, g_1 \oplus g_2 \oplus g_3 \oplus g_4\rangle$.

10.5 Formal mereology

Intuitively, the sum of some things is that which you get when you put them together. For many purposes, this rough intuition along with a quick glance at diagrams like the one in Figure 10.1 is sufficient. But if we want to prove that the logical representation of one sentence entails that of another sentence, and if these representations involve parthood or sums, then we need a precise framework in which we can substantiate our intuitions. For example, if we want to prove that (21b) logically follows from (21a), we need to show that this is the case given certain basic assumptions about the properties of parthood and sum.

The most commonly used framework for describing the formal behavior of parthood and sum in natural language seman-

tics is known as Classical Extensional Mereology (CEM). One of the advantages of CEM is that there are intuitive similarities between its parthood relation and set-theoretical subsethood, and between its sum operation and set-theoretical union. These similarities are listed in Table 10.1.

Table 10.1: Correspondences between CEM and set theory

	Property	CEM	Set theory
1	Reflexivity	$x \leq x$	$x \subseteq x$
2	Transitivity	$x \leq y \wedge y \leq z \rightarrow x \leq z$	$x \subseteq y \wedge y \subseteq z \rightarrow x \subseteq z$
3	Antisymmetry	$x \leq y \wedge y \leq x \rightarrow x = y$	$x \subseteq y \wedge y \subseteq x \rightarrow x = y$
4	Uniqueness	$P \neq \varnothing \rightarrow \exists! z\, \mathrm{sum}(z, P)$	$\exists! z\, z = \bigcup P$
5	Interdefinability	$x \leq y \leftrightarrow x \oplus y = y$	$x \subseteq y \leftrightarrow x \cup y = y$
6	Associativity	$x \oplus (y \oplus z) = (x \oplus y) \oplus z$	$x \cup (y \cup z) = (x \cup y) \cup z$
7	Commutativity	$x \oplus y = y \oplus x$	$x \cup y = y \cup x$
8	Idempotence	$x \oplus x = x$	$x \cup x = x$
9	Unique separation	$x \leq y \wedge x \neq y \rightarrow$ $\exists! z[x \oplus z = y \wedge$ $\neg \exists z'[z' \leq x \wedge z' \leq z]]$	$x \subset y \rightarrow \exists! z[z = y - x]$

There are different formulations of CEM. Here is a common one (for others see Hovda 2009). In the following, P is either an arbitrary predicate from first-order logic or an arbitrary set. Depending on the choice, the resulting system is first-order or second-order, because there are more sets than predicates.

The following axioms constrain parthood to be a partial order:

(33) **Axiom of reflexivity**
$\forall x[x \leq x]$
(Everything is part of itself.)

(34) **Axiom of transitivity**
$\forall x \forall y \forall z[x \leq y \wedge y \leq z \rightarrow x \leq z]$

(Any part of any part of a thing is itself part of that thing.)

(35) **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.)

Reflexivity is imposed on the parthood relation mainly for technical convenience. We can define an irreflexive *proper-part* relation by restricting parthood to nonequal pairs:

(36) **Definition: Proper part**
$x < y \stackrel{\text{def}}{=} x \leq y \wedge x \neq y$
(A proper part of a thing is a part of it that is distinct from it.)

With the part relation, we define the auxiliary concept of overlap:

(37) **Definition: Overlap**
$x \circ y \stackrel{\text{def}}{=} \exists z [z \leq x \wedge z \leq y]$
(Two things overlap if and only if they have a part in common.)

With this in place, we can define the notion of sum. We start by defining it in a way that leaves it open whether a collection of things may have more than one sum. It is only later that we will constrain the system so that no collection can have more than one sum.

(38) **Definition: Sum**
$\text{sum}(x, P) \stackrel{\text{def}}{=} \forall y [P(y) \rightarrow y \leq x] \wedge \forall z [z \leq x \rightarrow \exists z' [P(z') \wedge z \circ z']]$
(A sum of a set P is a thing that contains everything in P and whose parts each overlap with something in P.)

Here, sum is a relation of type $\langle e, \langle \langle e, t \rangle, t \rangle \rangle$, that is, it relates entities x of type e to predicates P of type $\langle e, t \rangle$. The formulation of the definition reflects the intuitive fact that a sum may have other

parts than just its immediate components. For example, the sum of (i) the referent of the conjoined term *John and Mary* and (ii) the referent of the proper name *Bill* has more parts than these two individuals: for example, it also has the referent of *John* among its parts, as well as the referent of *Bill and Mary*.

The following facts follow from these definitions:

(39) **Fact**
$\forall x \forall y [x \leq y \rightarrow x \circ y]$
(Parthood is a special case of overlap.)

(40) **Fact**
$\forall x [\text{sum}(x, \{x\})]$
(A singleton set sums up to its only member.)

Different systems of mereology disagree on what kinds of collections have a sum, and whether it is possible for one and the same collection to have more than one sum. In CEM, sums are unique, therefore two things composed of the same parts are identical. This is expressed by the following axiom:

(41) **Axiom of uniqueness of sums**
$\forall P [P \neq \varnothing \rightarrow \exists! z \, \text{sum}(z, P)]$
(Every nonempty set has a unique sum.)

The binary and generalized sum operators in (42) and (43) give us a way to refer explicitly to the sum of two things, and to the sum of an arbitrary set. In these expressions, $\iota x \, P(x)$ is defined if and only if there is exactly one object x such that $P(x)$ is true. When defined, the expression denotes that object.

(42) **Definition: Binary sum**
$x \oplus y$ is defined as $\iota z \, \text{sum}(z, \{x, y\})$.
(The sum of two things is the thing which contains both of them and whose parts each overlap with one of them.)

(43) **Definition: Generalized sum**

> For any nonempty set P, its sum $\bigoplus P$ is defined as $\iota z\,\mathrm{sum}(z, P)$. (The sum of a set P is the thing which contains every element of P and whose parts each overlap with an element of P.)

For example, we can write the plural individual denoted by the conjoined term *John and Mary* as "$j \oplus m$", and to the sum of all water as "$\bigoplus$water". Since we can refer to this sum as *the water*, the $\bigoplus$ operator can be thought of as a somewhat similar to a definite description.

We can now prove that various entailment relations hold between sentences that we had represented in ways that look rather different from each other. For example, we can prove that the assumption $\mathsf{boy(j)} \wedge \mathsf{boy(b)}$ entails the conclusion $^{*}\mathsf{boy(j \oplus b)}$. According to Definition (18), $^{*}\mathsf{boy}$ is the set that contains any sum of any nonempty set of boys. So, $^{*}\mathsf{boy(j \oplus b)}$ is true if and only if $\mathsf{j \oplus b}$ is the sum of some nonempty set of boys. The obvious candidate is $\{\mathsf{j}, \mathsf{b}\}$. So we need to show two things: that $\{\mathsf{j}, \mathsf{b}\}$ is a nonempty set of boys, and that $\mathsf{j \oplus b}$ is its sum. By assumption, $\mathsf{boy(j)} \wedge \mathsf{boy(b)}$, hence j and b are boys. So $\{\mathsf{j}, \mathsf{b}\}$ is a nonempty set of boys. That $\mathsf{j \oplus b}$ is the sum of this set follows from the definition of $\oplus$. This concludes the proof.

10.6 A formal fragment

Finally, we need to extend the syntax and semantics of our logic in order to adapt it to the new entities and relations we have added. The syntax is defined as in three-valued type logic (L_{λ} with three truth values as in Chapter 8), plus the following additions:

10.6.1 Logic syntax

We add the following primitive symbol to our syntax.

1. **Parthood** If α and β are terms of type e, then $\alpha \leq \beta$ is an

expression of type t.

In addition, we have the following abbreviation conventions.

1. If ϕ is an expression of type $\langle e,t\rangle$, we write $\bigoplus\phi$ (read as: "the sum of ϕ") for the expression $[\iota x.[\forall y.[\phi(y) \rightarrow y \leq x]] \wedge [\forall z.[\forall z'.[\phi(z') \rightarrow z' \leq z]] \rightarrow x \leq z]]$. (That is, $\bigoplus\phi$ denotes the least upper bound, or sum, of ϕ – see Definition **??**. By Axiom **??**, this will be defined whenever ϕ applies to at least one entity.)

2. If α and β are terms of type e, we write $[\alpha \oplus \beta]$ (read as: "the sum of α and β") for the expression $\bigoplus[\lambda x.x = \alpha \vee x = \beta]$.

3. An expression of the form $[[\alpha \oplus \beta] \oplus \gamma]$ or $[\alpha \oplus [\beta \oplus \gamma]]$ can be simplified to $[\alpha \oplus \beta \oplus \gamma]$.

10.6.2 Logic semantics

Expressions are interpreted with respect to both:

- a model $M = \langle D, I, \leq\rangle$ where D and I are defined as usual and $\leq$ is the parthood relation over individuals that obeys the conditions listed above,

- an assignment g defined as usual.

For every well-formed expression α, $[\![\alpha]\!]^{M,g}$, is defined recursively as usual. We add the following rule:

(44) **Parthood**
If α and β are expressions of type e, then $[\![\alpha \leq \beta]\!] = 1$ if $[\![\alpha]\!] \leq [\![\beta]\!]$, otherwise $[\![\alpha \leq \beta]\!] = 0$.

10.6.3 English syntax

Syntax rules. We add the following rules for coordination:

(45) **Syntax**

X	$\rightarrow$	X CoordP
CoordP	$\rightarrow$	Coord X

where $X \in \{\text{S, VP, DP, V,}\ldots\}$

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*

10.6.4 Translations

Type $\langle e, t\rangle$:

1. *smokes* $\rightsquigarrow \lambda x.{}^*\mathsf{Smoke}(x)$
2. *drinks* $\rightsquigarrow \lambda x.{}^*\mathsf{Drink}(x)$
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. *Mary* $\rightsquigarrow$ m
3. *Tom* $\rightsquigarrow$ t
4. *Dick* $\rightsquigarrow$ d
5. *Harry* $\rightsquigarrow$ h

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. $[\mathsf{Smokes(j)} \wedge \mathsf{Drinks(m)}]$
b. $\mathsf{Smokes(j)}$

Every model for (2a) is also a model for (2b).

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 represent (3a) given the previous chap-

ters (we are treating *the toast* as if it was a constant rather than a definite description, but nothing will hinge on this):

(4) $\mathsf{Butter(j,t)}$

If this representation is correct, (3a) is about only two entities: Jones and the toast. Which entity does *slowly* describe in (3a)? Is it perhaps Jones who is slow? Then we might represent the denotation of that sentence as follows:

(5) $[\mathsf{Butter(j,t)} \wedge \mathsf{Slow(j)}]$

Since (5) logically entails (4), we have an account of the entailment from (3a) to (3b). But there is a problem. If we represent (3a) as (5), clearly we ought to represent (6a) as (6b), by analogy.

(6) a. Jones buttered the bagel quickly.
b. $[\mathsf{Butter(j,b)} \wedge \mathsf{Quick(j)}]$

But then, in any model where (5) and (6b) are both true, the following will also be true as a matter of logical consequence!

(7) $[\mathsf{Slow(j)} \wedge \mathsf{Quick(j)}]$

Unless we want to countenance the possibility that Jones is both slow and quick at the same time, our account clearly has a problem.

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, 1990):

(8) 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, where e ranges over events:

(9) a. $\exists e.[\mathsf{Butter}(\mathsf{j},\mathsf{t},e) \wedge \mathsf{Slow}(e)]$
b. $\exists e.[\mathsf{Butter}(\mathsf{j},\mathsf{b},e) \wedge \mathsf{Quick}(e)]$

A sentence like (3b) would then be represented as:

(10) $\exists e.\mathsf{Butter}(\mathsf{j},\mathsf{t},e)$

There is a logical entailment from (9a) to (10), as desired. But unlike before, the conjunction of (9a) and (9b) 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:

(11) 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:

(12) 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. $\exists e.\mathsf{Butter}(\mathsf{j},\mathsf{t},e)$

Perceptual reports. Since events are concrete entities with a location in spacetime, it stands to reason that we can see and hear them. This idea can be exploited to give semantics of direct perception reports (Higginbotham, 1983):

(13) a. John saw Mary leave.
b. ∴ Mary left.

(14) a. $\exists e \exists e'.\mathsf{Saw}(\mathsf{j},e,e') \wedge \mathsf{Leave}(\mathsf{m},e')$
b. $\exists e'.\mathsf{Leave}(\mathsf{m},e')$

Here, e is the event of John seeing, and e' is the event that John sees—that is, the event of Mary leaving.

The relation between adjectives and adverbs. If adverbs ascribe properties to events, it is plausible to assume that the same is true of adjectives that are derivationally related to these adverbs (Parsons, 1990):

(15) a. Brutus stabbed Caesar violently.
b. ∴ There was something violent.

(16) 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, 1990). The relationship between this event and syntactic arguments of the verb is then expressed by a smallish number of semantic relations with names like AGENT, THEME, INSTRUMENT, and BENEFICIARY. These relations represent ways entities take part in events and are generally called THEMATIC ROLES. 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 disadvantage 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 this is what we will do 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:

(17) $\exists e.\mathsf{Butter}(e) \wedge \mathsf{agent}(e,\mathsf{j}) \wedge \mathsf{theme}(e,\mathsf{t})$

In Neo-Davidsonian event semantics, there is no fundamen-

tal 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. As in the case of *the toast*, this is not essential.)

(18) $\exists e.\mathsf{Butter}(e) \wedge \mathsf{agent}(e,\mathsf{j}) \wedge \mathsf{theme}(e,\mathsf{t}) \wedge \mathsf{instr}(e,\mathsf{k})$

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.

(19) 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:

(20) $\exists e.\mathsf{Load}(e) \wedge \mathsf{agent}(e,\mathsf{j}) \wedge \mathsf{goal}(e,\mathsf{t}) \wedge \mathsf{theme}(e,\mathsf{h})$

The argument *the truck* in (19a) parallels the adjunct *onto the truck* in (19b), and the adjunct *with the hay* in (19a) parallels the argument *the hay* in (19b).

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, the number of semantics

arguments of a verb is fixed.

(21) 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 *eat*, *dine*, and *devour* that could explain why the first is optionally and the second is obligatorily intransitive while the third is obligatorily transitive. Whatever distinction there is between them must arguably instead be attributed to syntax.

One of the advantages of the Neo-Davidsonian view is that it allows us to capture semantic entailment relations between different syntactic subcategorization frames of the same verb, such as causatives and their intransitive counterparts (Parsons, 1990):

(22) a. Mary felled the tree.
b. ∴ The tree fell.

(23) a. $\exists e.\mathsf{Fall}(e) \wedge \mathsf{agent}(e,\mathsf{m}) \wedge \mathsf{theme}(e,\mathsf{t})$
b. $\exists e.\mathsf{Fall}(e) \wedge \mathsf{theme}(e,\mathsf{t})$

(24) a. Mary opened the door.
b. ∴ The door opened.

(25) 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 (26a) – perhaps beneficiary – matches the one of *Mary* in (26b).

(26) 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 (26a), but in (26b) 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 correspond to (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 **?** 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 corresponds to exactly one role, or whether the subject of a verb like *fall* is both the agent and the theme of the event (Parsons, 1990). Relatedly, it is often assumed that each event has at most one agent, at most one theme, and so on. This view, often called the *unique role requirement* or *thematic uniqueness*, is widely accepted in semantics (Carlson, 1984; Parsons, 1990; Landman, 2000). Thematic uniqueness has the effect that thematic roles can be represented as partial functions. This is often reflected in the notation, as in (27).

(27) $\exists e.\,\mathsf{Butter}(e) \wedge \mathsf{agent}(e) = \mathsf{j} \wedge \mathsf{theme}(e) = \mathsf{t}$

A differing, less com 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 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, 1990, 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.

11.2.1 Verbs as predicates of events

On the first strategy, verbs denote predicates of events:

(28) 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.

(29) **Syntax**
DP $\rightarrow$ θ DP

We then write lexical entries that map these heads to suitable roles:

(30) **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:

(31) 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.

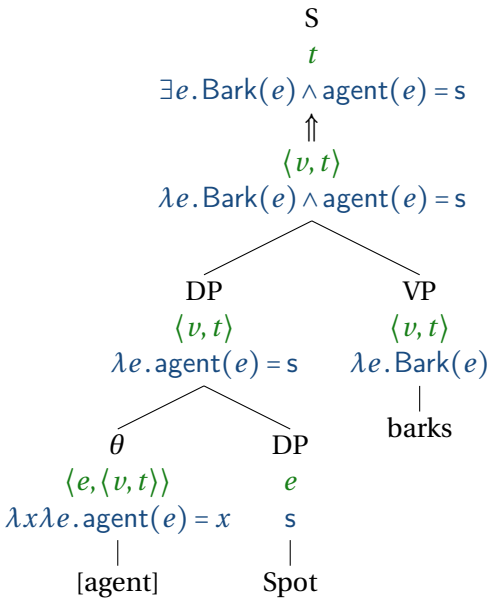
Type-Shifting Rule 5. Existential closure
If $\alpha \rightsquigarrow \alpha'$, where α' is of category $\langle v, t\rangle$, then:

$$\alpha \rightsquigarrow \exists e.\, \alpha'(e)$$

as well (as long as e does not occur in α'; in that case, use a different variable of the same type).

A sample derivation that shows all of the elements we have introduced is shown in (32). 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.

(32)



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:

(33) **Syntax**

S	→	AdvP S
VP	→	AdvP VP
VP	→	VP AdvP
AdvP	→	Adv

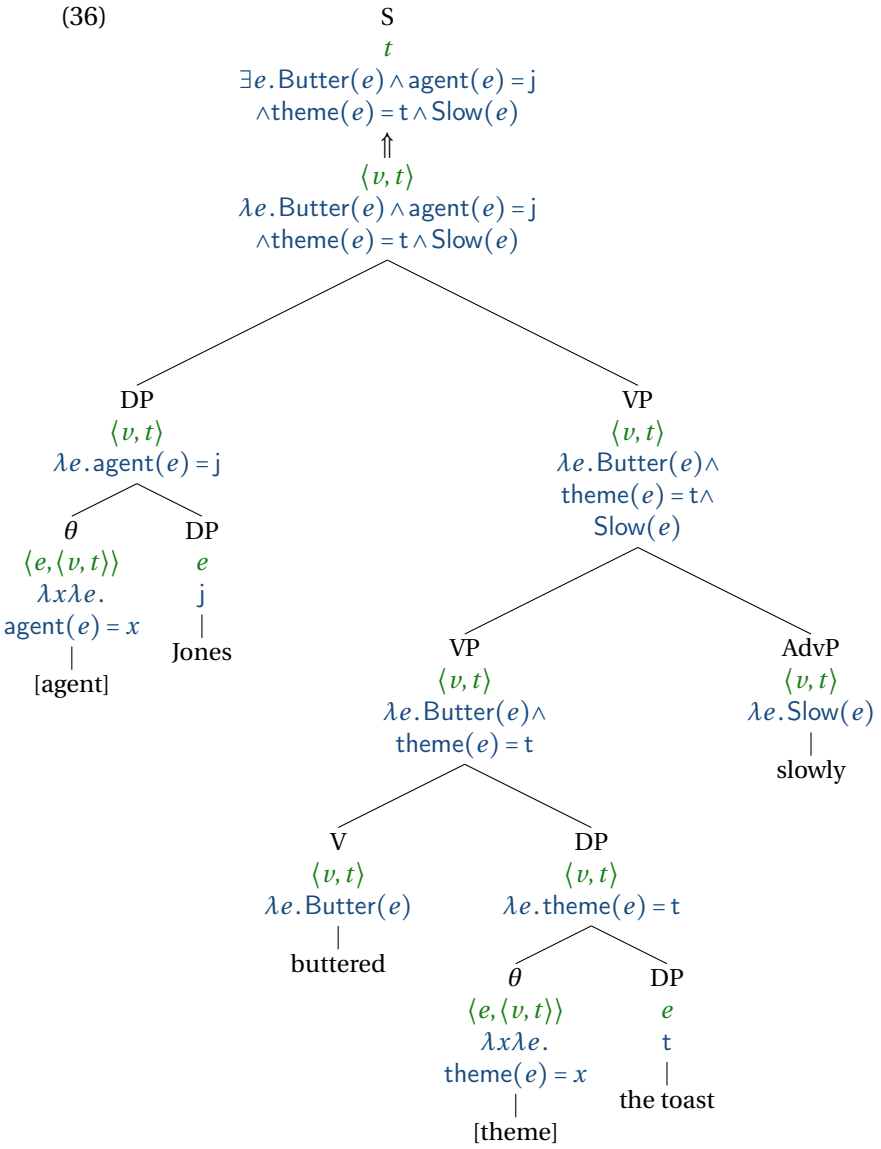
(34) **Lexicon**
Adv: slowly

As we have seen above, *slowly* is interpreted as an event predicate. Its lexical entry is therefore very simple:

(35) a. *slowly* $\rightsquigarrow \lambda e.\mathsf{Slow}(e)$

The tree in (36) 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:

(36)



In the derivation in (36), 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.

11.2.2 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:

(37) **Syntax**
DP $\rightarrow$ θ DP

Lexicon. Lexical items are associated with syntactic categories as follows:

θ: [agent], [theme], ...

Translations. Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- e is $v_{0,v}$
- Bark and Butter are constants of type $\langle v, t\rangle$,
- agent and theme are constants of type $\langle v, e\rangle$.

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]* $\rightsquigarrow \lambda x \lambda e.\mathsf{theme}(e) = x$

11.3 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 (38), read with neutral intonation, is not ambiguous. Its only reading corresponds to (39b), where the event quantifier takes low scope). As for (40b), that is not a possible reading of the sentence.

(38) No dog barks.

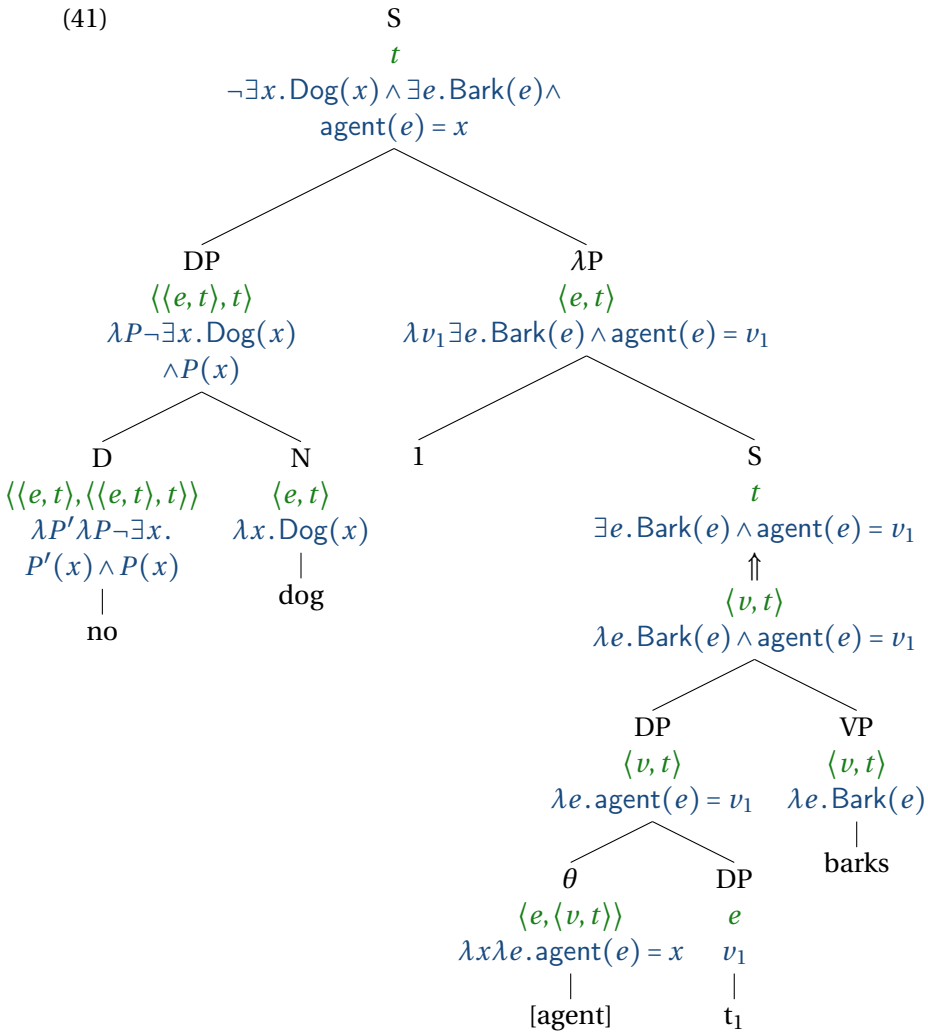
(39) a. $\neg[\exists x.\mathsf{Dog}(x) \wedge \exists e.\mathsf{Bark}(e) \wedge \mathsf{agent}(e) = x]$
b. "There is no barking event that is done by a dog"

(40) 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"

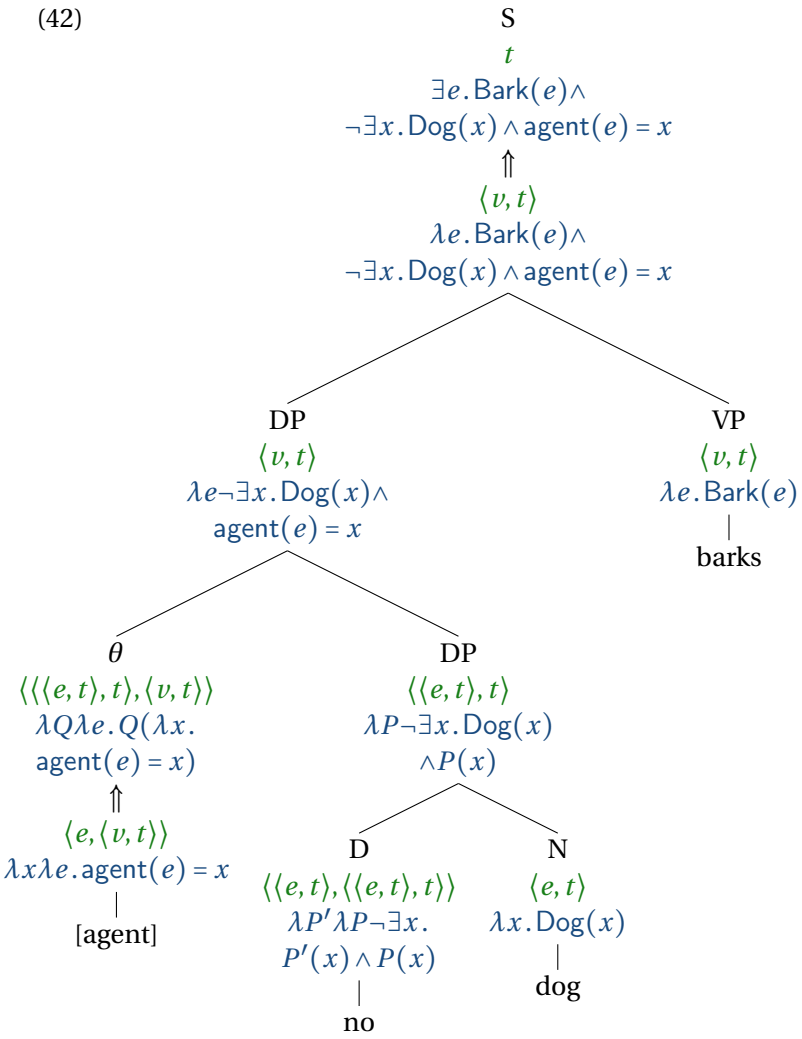
Exercise 1. How can you tell that (40b) is not a possible reading of sentence (38)?

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 (39b), while applying Hendriks' object raising rule (or rather, the general schema) to the theta role head leads to the unavailable reading (40b). This is shown in (41) and (42), respectively.

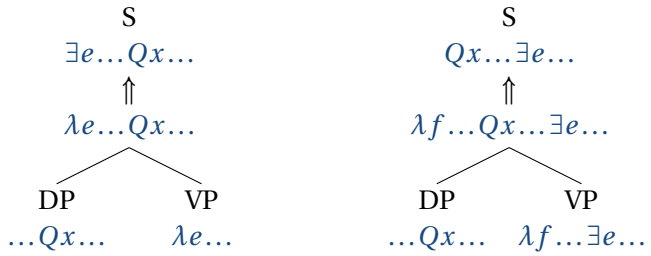
(41)


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
$\exists e.\mathsf{Bark}(e) \wedge \mathsf{agent}(e) = v_1$
⇑
$\langle v,t\rangle$
$\lambda e.\mathsf{Bark}(e) \wedge \mathsf{agent}(e) = v_1$
DP
$\langle v,t\rangle$
$\lambda e.\mathsf{agent}(e) = v_1$
VP
$\langle v,t\rangle$
$\lambda e.\mathsf{Bark}(e)$
barks
θ
$\langle e,\langle v,t\rangle\rangle$
$\lambda x \lambda e.\mathsf{agent}(e) = x$
[agent]
DP
e
v_1
t_1

(42)



The interim conclusion, then, is that event semantics seems to commit us to a quantifier-raising based treatment of quantificational noun phrases.



(a) $\exists e$ introduced at S (b) $\exists e$ introduced by verb

Figure 11.1: Comparison of two approaches to event semantics

11.3.1 Verbs as event quantifiers

In the tree in (41), 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. 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.

To implement this approach, we need to revise our semantics. We will equip each verb with a variable f of type $\langle v, t \rangle$, which 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 lex-

ical 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:

(43) 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:

(44) 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:

(45) 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. $\exists e[\mathsf{Bark}(e) \wedge \mathsf{ag}(e) = \mathsf{s}]$

To formalize this idea, we introduce the type-shifting rule of Quantifier Closure:

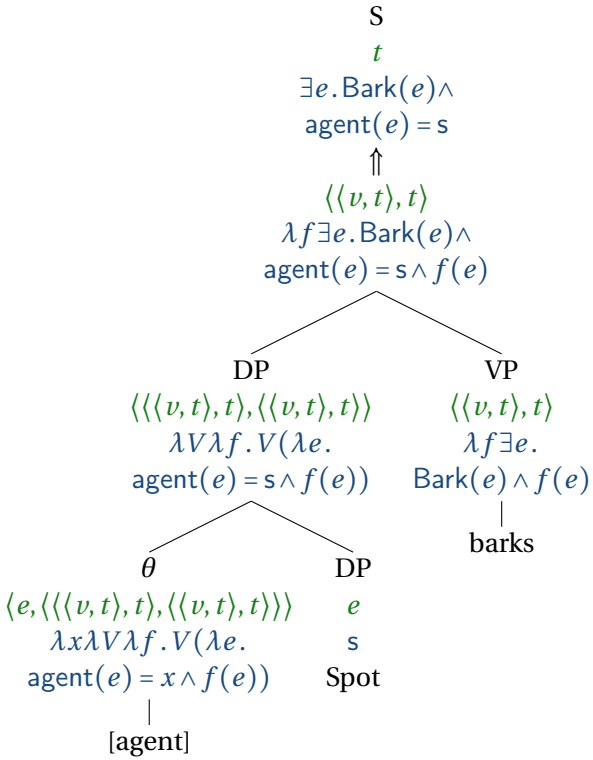
Type-Shifting Rule 6. Quantifier Closure
If $\alpha \rightsquigarrow \alpha'$, where α' is of category $\langle\langle v, t\rangle, t\rangle$, then:

$$\alpha \rightsquigarrow \alpha'(\lambda e.\mathsf{true})$$

as well.

The full derivation of the sentence is shown in (46).

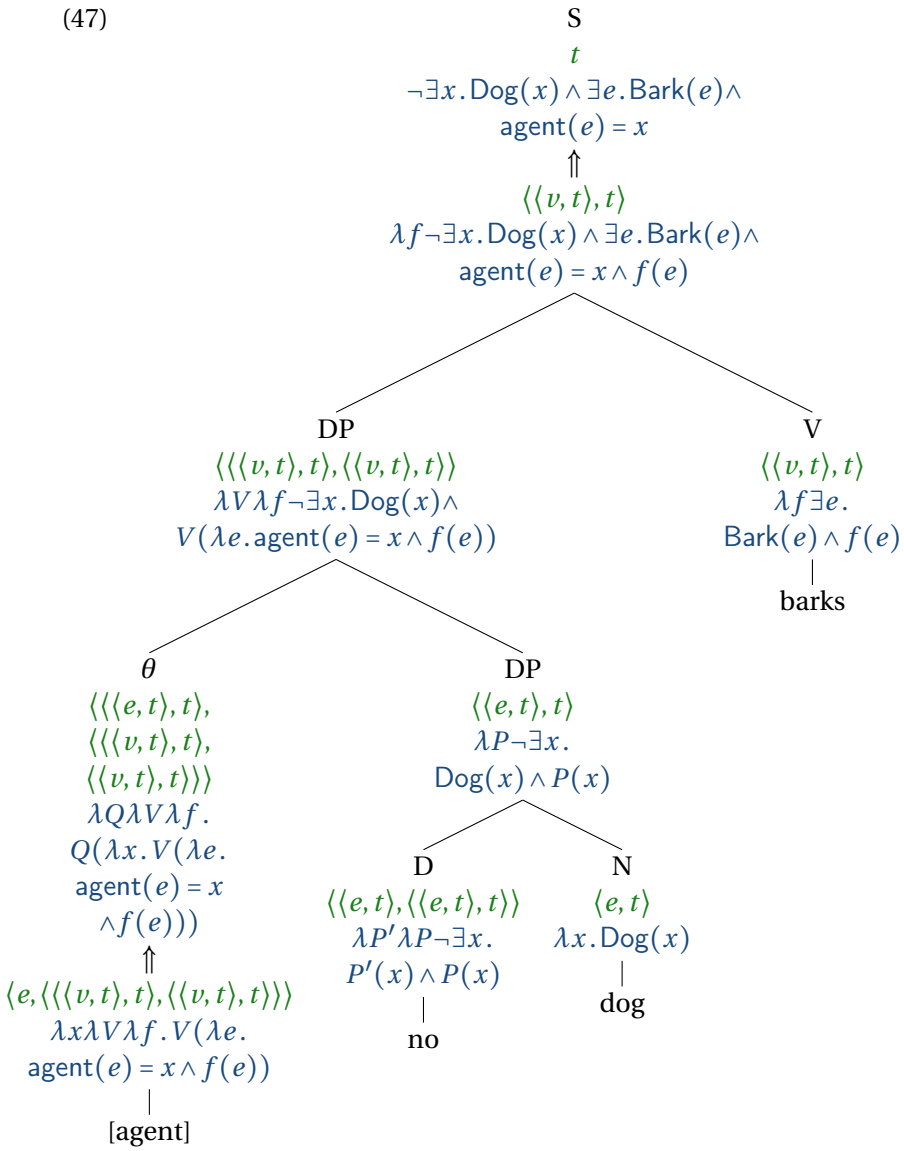
(46)



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 (47). We do not need to apply quan-

tifier 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.

(47)



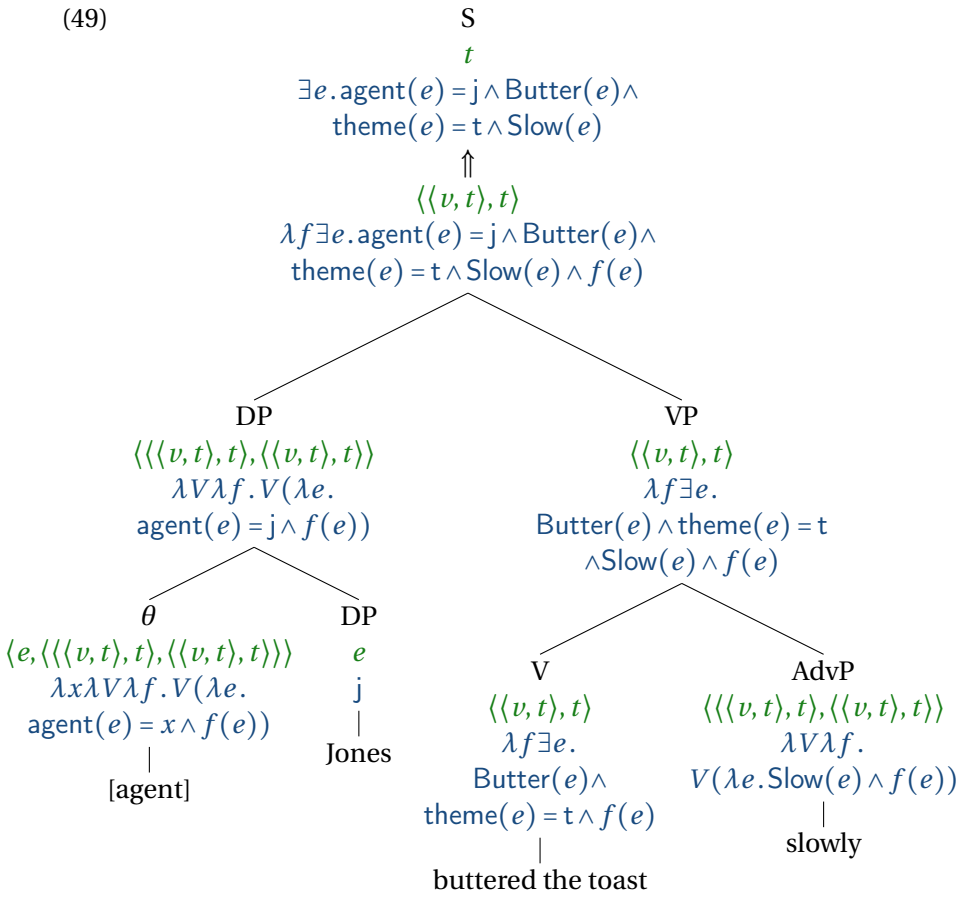
Let us now see how syntactic adjuncts, such as adverbs, are treated on this approach. Just like syntactic arguments, adjuncts

are combined with verbal projections using Function Application instead of Predicate Modification. This makes the representations of adverbs more complicated:

(48) 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 (49). 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.

(49)



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.

11.3.2 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:

(50) **Syntax**
DP → θ DP

Lexicon. Lexical items are associated with syntactic categories as follows:

θ: [agent], [theme], ...

Translations. Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- e is $v_{0,v}$,
- 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.

11.4 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:

(51) $$\text{and}_{\langle \tau, \langle \tau, \tau \rangle \rangle} \rightsquigarrow \begin{cases} \lambda q \lambda p . p \wedge q & \text{if } \tau = t \\ \lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1} . \langle\!\langle \text{and} \rangle\!\rangle_{\langle \sigma_2, \langle \sigma_2, \sigma_2 \rangle \rangle}(X(Z))(Y(Z)) & \text{if } \tau = \langle \sigma_1, \sigma_2 \rangle \end{cases}$$

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 $\tau = \langle v, \langle v, t \rangle \rangle$. Applying rule (51) in each case results in the following:

(52) a. $\text{and}_{\text{VP}} \rightsquigarrow \lambda f' \lambda f \lambda e . f(e) \wedge f'(e)$
b. $\text{and}_{\text{VP}} \rightsquigarrow \lambda V' \lambda V \lambda f . V(f) \wedge V'(f)$

Exercise 2. Show how rule (51) leads to these two representations.

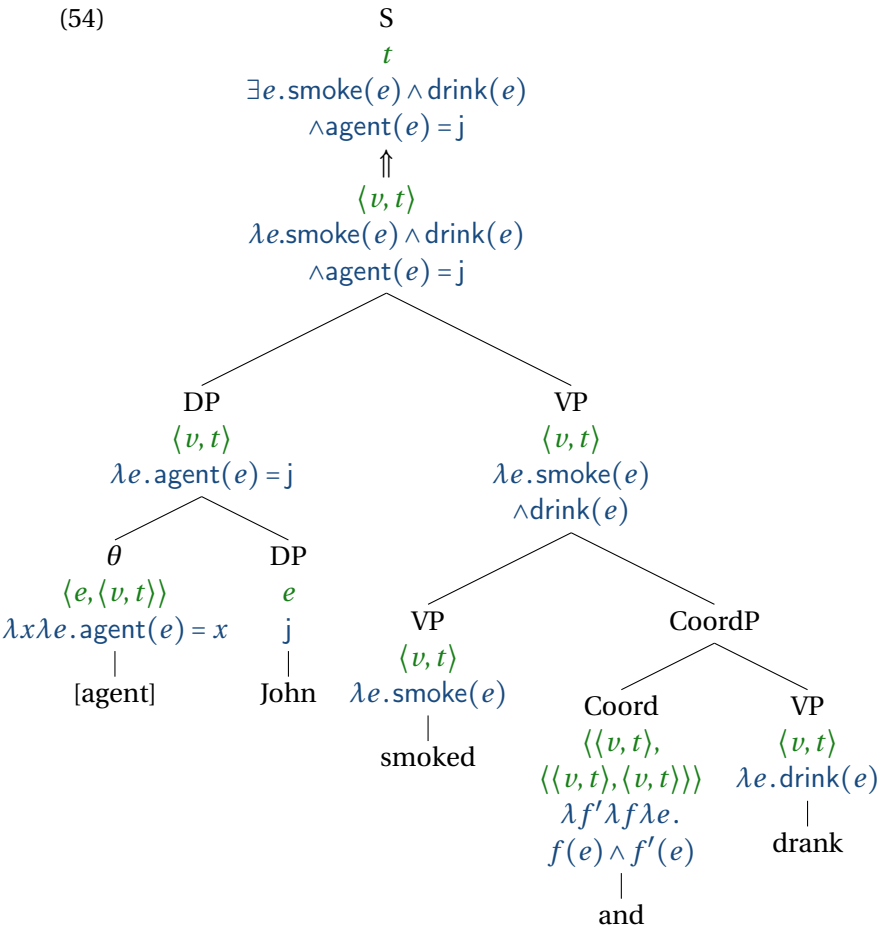
As you can see in (54) and (55), these two choices lead to very different translations: (53a) and (53b) respectively.

(53) a. $\exists e . \mathsf{smoke}(e) \wedge \mathsf{drink}(e) \wedge \mathsf{agent}(e) = \mathsf{j}$
b. $\exists e . \mathsf{smoke}(e) \wedge \mathsf{agent}(e) = \mathsf{j} \wedge$
$\exists e' . \mathsf{drink}(e') \wedge \mathsf{agent}(e') = \mathsf{j}$

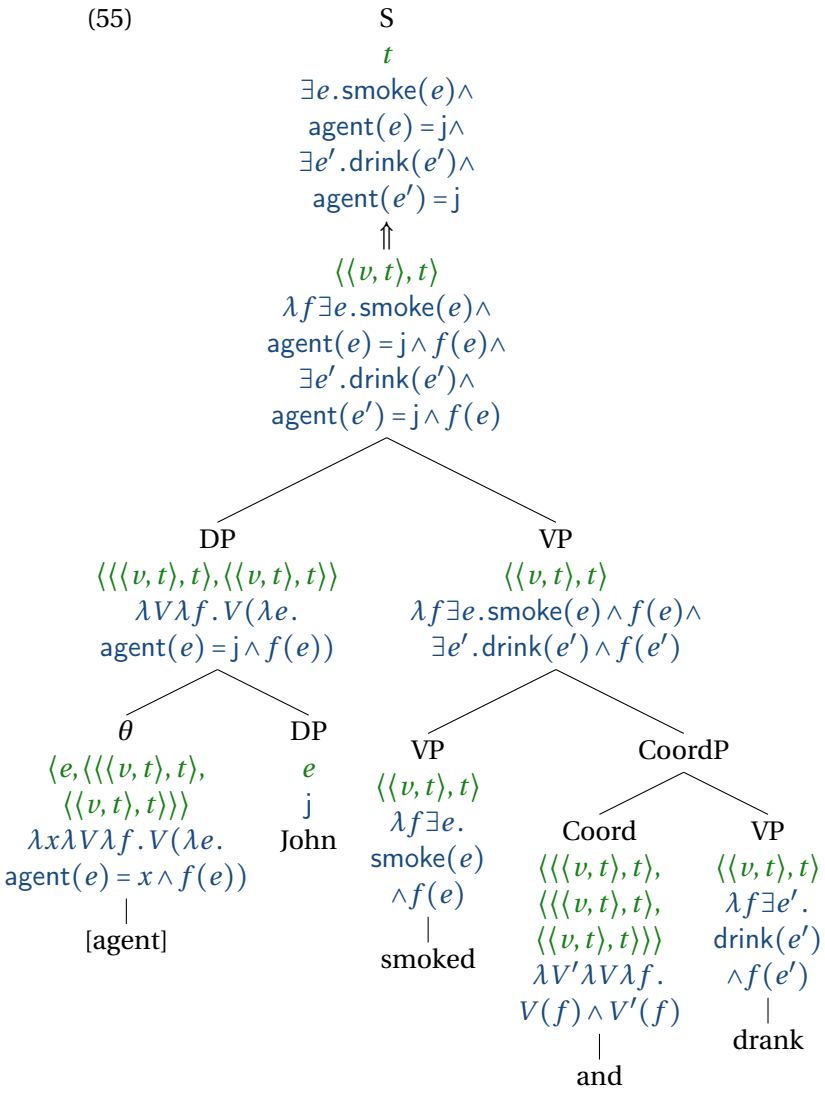
Now, (53a) cannot be the right representation of *John smoked and drank*. If this sentence is true, for all we know he might have smoked slowly and drunk quickly. In (53a) there is only one event for two such contradictory adverbs to modify, and we would end up with the same kind of problem we already 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 (53a) sends us right back to square one. We fare much better with (53b), because it provides us with the two events we need to avoid the problem.

Exercise 3. Add *slowly* and *quickly* to the tree in (55) and show how the resulting formula avoids the attribution of contradictory properties to the same event.

(54)


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
$\langle v, t \rangle$
$\lambda e.\mathsf{agent}(e) = \mathsf{j}$
VP
$\langle v, t \rangle$
$\lambda e.\mathsf{smoke}(e)$
$\wedge \mathsf{drink}(e)$
θ
$\langle e, \langle v, t \rangle \rangle$
$\lambda x \lambda e.\mathsf{agent}(e) = x$
[agent]
DP
e
j
John
VP
$\langle v, t \rangle$
$\lambda e.\mathsf{smoke}(e)$
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

(55)



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 predi-

cates. 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, $\theta(e \oplus e') = \theta(e) \oplus \theta(e')$.

11.5 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. In Section **??**, we have analyzed negation as an operator that takes scope at VP, between the subject and the rest of the sentence, corresponding to its surface position in English. (We have also considered the possibility that negation applies at the level of the sentence. Here we leave this possibility aside.)

Negation, just like quantificational noun phrases, always seems to take scope above the event quantifier. For example, (56), read with neutral intonation, only has the reading in (57b), and lacks the reading in (58b). 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.

(56) Spot didn't bark.

(57) a. $\neg[\exists e.\mathsf{Bark}(e) \wedge \mathsf{agent}(e) = s]$
b. "There is no barking event that is done by Spot"

(58) 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 fare that we

have encountered? 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:

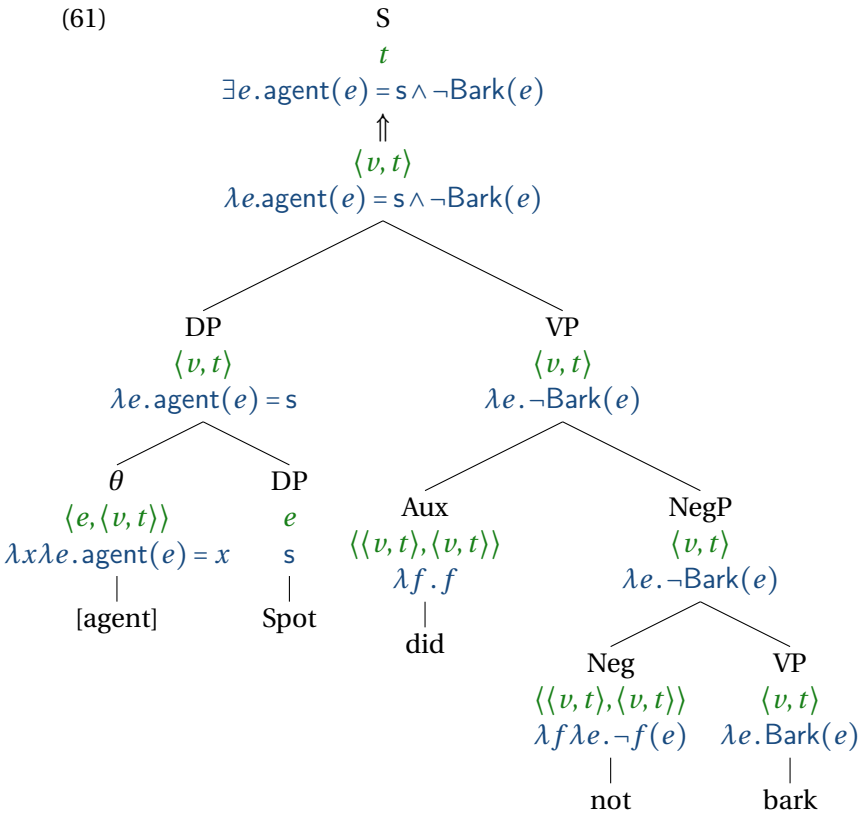
(59) not $\rightsquigarrow \lambda f \lambda e . \neg f(e)$ (to be revised)

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 (61). The result is the following reading:

(60) $\exists e . \mathsf{agent}(e) = \mathsf{s} \wedge \neg \mathsf{Bark}(e)$
There is an event whose agent is Spot that is not a barking event.

This is much weaker than what we want. It 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 (59). 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.

(61)

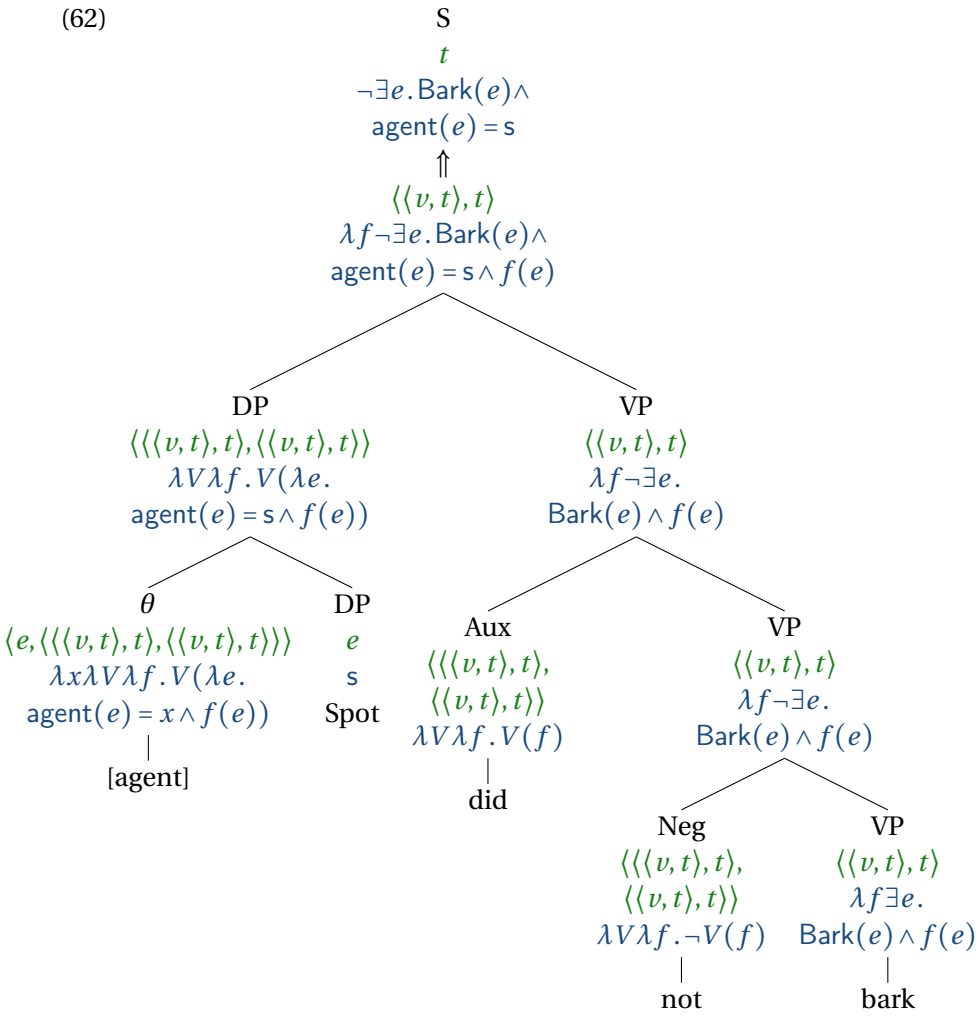


One can respond to this situation in different ways. One way is to expand our inventory of events to include "negative events". In some cases, it is intuitively clear what a negative event should be: for example, a negative staying event is a leaving event and vice versa. While it is not so clear what 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.

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 (58b). This is shown in Figure (62).

(62)


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

So far, we have been ignoring the contribution of tense. In this chapter, we will finally face it. In order to do so, we must grapple with the related issue of aspect. Ideally, our theory should be able to explain the contrasts in meaning among all of the following forms.

(1) a. Ann dances. [simple present]
b. Ann danced. [simple past]
c. Ann will dance. [simple future]

(2) a. Ann is dancing. [present progressive]
b. Ann was dancing. [past progressive]
c. Ann will be dancing. [future progressive]

(3) a. Ann has danced. [present perfect]
b. Ann had danced. [past perfect]
c. Ann will have danced. [future perfect]

(4) a. Ann has been dancing. [present perfect progressive]
b. Ann had been dancing. [past perfect progressive]
c. Ann will have been dancing. [future perfect prog.]

We begin with aspect, in both of its senses, and then move on to tense.

12.2 Aspect

12.2.1 Aktionsart

The term ASPECT can refer to two different things in linguistic theory. Both have to do with the temporal properties of a state or event being described or referred to. One of these two things is also called AKTIONSART, a German word that literally means 'type of action'. In a famous paper entitled *Verbs and Times*, Vendler (1957) distinguished between four types of eventualities:

- States (example: *have a tan*) 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: *jump*). They are like achievements except they do not lead to a change. These 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:

(5) a. Ida ran a mile in an hour. [accomplishment]

[1] Other words for 'aktionsart' include *lexical aspect, situation aspect, internal aspect, aspectual class,* and *situation type.*

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*:

(6) 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 requires energy, while having a tan does not. The state/non-state distinction also has reflexes in the grammar. The progressive in English does not combine well with stative predicates:

(7) a. Ida is running along the beach. [activity]
b. ??Ida is having a tan. [state]

Furthermore, the simple present tense gives rise to a habitual interpretation only with non-stative predicates:

(8) a. Ida runs along the beach. [activity: habitual]
b. Ida has a tan. [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:

(9) 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:

(10) a. Ida jumped for an hour. [semelfactive: iterative]
b. ??Ida reached the pier for an hour. [achievement]

The idea of (10a) 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.

12.2.2 Viewpoint aspect

We turn now to the other kind of aspect, which goes by many names, including *viewpoint aspect*, *grammatical aspect*, and *perspective point*. We choose the term VIEWPOINT ASPECT here, em-

phasizing the idea that it has to do with how an eventuality is 'viewed', not with its inherent temporal properties. According to a prominent view on viewpoint aspect (Klein, 1994), this kind of aspect provides a link between eventualities and tense, by specifying the relation between the EVENT TIME and a REFERENCE TIME, two concepts which we will explain shortly.

English has two morphological forms that express viewpoint aspect: the perfect, as in *I have eaten*, and progressive, as in *I am eating*. Confusingly enough, the English progressive expresses IMPERFECTIVE ASPECT. Stative predicates like *have a tan* are also imperfective. So the two main aspectual distinctions that English is sensitive to are perfect vs. non-perfect, and perfective vs. imperfective. Confusingly enough, 'perfective' is totally different from 'perfect'; these categories can cross-classify:

	PERFECTIVE	IMPERFECTIVE
PERFECT	*I have danced*	*I have been dancing*
NON-PERFECT	*I danced*	*I was dancing*

The English perfect is a key motivation for Reichenbach's (1947) theory of temporal reference. Reichenbach noticed 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 (later called TOPIC TIME).[2]

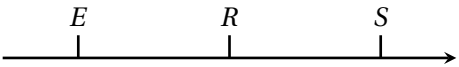
- SPEECH TIME (*S*): the time the sentence is uttered
- EVENT TIME (*E*): the time the event takes place
- REFERENCE TIME (*R*): the time under discussion (also known as *Topic Time*)

The concept of 'reference time' was Reichenbach's major innovation, and the concept that is least intuitively obvious. One way of

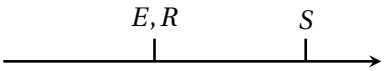
[2] This section borrows quite liberally from Cable's (2008) notes on tense.

characterizing it is that it is the time that the sentence is 'about' (hence the term 'topic time').

According to Reichenbach, the difference between a sentence like *I had danced* and a sentence like *I danced* is that in the former case, with the perfect, the sentence is about a time prior to speech time before which some dancing took place (so $E < R < S$):



In the case of the simple past, *I danced*, the sentence is about the time at which the dancing 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:

(11) a. At 5pm, I danced. (5pm = dancing time)
b. At 5pm, I had danced. (dancing time < 5pm)

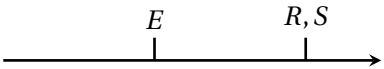
Assuming the modifier is identified with the reference time, we correctly predict that 5pm is the time of dancing for the sentence in simple past, but a time before the speech time and after the time of dancing in the past perfect.

The contribution of the perfect, then, is that the event time precedes the reference time. This holds in other tenses as well. Consider the present tense:

(12) a. I have danced.
b. I dance.

The simple present example in (12b) has a habitual present interpretation; the present perfect example expresses that a dancing

event took place prior to speech time. This can be understood if the present tense puts reference time at speech time, and the contribution of the perfect is to locate the event time as being prior to reference time. The present perfect thus looks like this:



And the simple present looks like this:

This theory of the perfect works in the future tense as well. If we assume that the future tense locates the reference time after speech time, and the perfect locates the event time prior to the reference time, a sentence like:

(13) At 5pm, I will have danced.

is predicted to imply that the time referred to by 5pm is in the future, and that some dancing event will have occurred prior to that (perhaps before, perhaps after speech time). This accords with intuition. In contrast:

(14) At 5pm, I will dance.

implies that the time referred to by 5pm is in the future, and that a dancing event will take place at that time.

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 viewpoint aspect is to relate the event time to the reference time. Past tense locates the reference time prior to speech time; present tense sets them equal; and future locates reference time after speech time. The perfect places the event prior to reference time; otherwise the event takes place at reference time.

This view is somewhat of an oversimplification; there are a number of different uses for the English perfect, including:

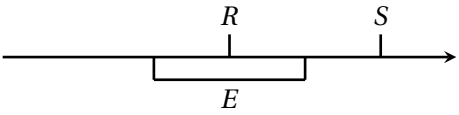
(15) a. Ed has put the cake in the oven. RESULTATIVE
b. Ed has visited Korea many times. EXISTENTIAL
c. Ed has lived in Korea for 3 years. UNIVERSAL

We will set these uses aside.

Let us now turn to perfective and imperfective aspect, which also relate the event time to the reference time. Consider the following contrast:

(16) a. At 5pm, I danced.
b. At 5pm, I was dancing.

In the first case, there was a dancing event that took place at 5pm. In the second case, there was a dancing event that extended across 5pm. Klein (1994) proposes to model this using inclusion among time intervals. In the past progressive example (16b), the time interval during which the dancing event took place includes the reference time interval identified with *5pm*. This gives us the following picture for past progressive:



In general, imperfective aspect signals that the event time contains the reference time, while perfective aspect signals the reverse: 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: $E \subseteq R$
- imperfective aspect: $R \subseteq E$

Stative predicates also have imperfective aspect:

(17) At 5pm, I was asleep.

As with the progressive, the time of the sleeping eventuality includes the reference time. Thus progressive is a specific type of imperfective.

To summarize, tense relates speech time with reference time, and aspect relates reference time with event time. In the following sections, we will work towards a formal and compositional implementation of these ideas, which will require us to both establish the machinery for talking about indexicality and make certain decisions about exactly how to consider reference time: Is it more like an existentially bound variable or more like a free variable? We turn first to the issue of indexicality.

12.3 Indexicality

The notion of 'speech time' is an INDEXICAL one: It has to do with the so-called CONTEXT OF UTTERANCE, or CONTEXT OF USE. In Chapter 9 on dynamic semantics, we spoke of context in terms of the information that has been established so far in the discourse, and the discourse referents that had been introduced, and how denotations could be seen as operations that update such contexts. The 'context of utterance' is a context in another sense: It's about who is speaking, to whom, where, when, etc.

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]".[3]

To warm up intuitions regarding how indexicals should be analyzed, consider the following two utterances:

(18) 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.

What do you think: Do they have the same meaning or different meaning? How about the following pair:

(19) 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.

It's not immediately obvious how to answer this question. In some sense, the examples in (18) have the same meaning, but in another sense, those in (19) have the same meaning.

Kaplan resolves this tension by distinguishing between two levels of meaning, CHARACTER and CONTENT.

- CHARACTER is the aspect of meaning that two utterances of the same sentence share across different contexts of utterance.

[3]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.

- CONTENT is the proposition expressed by an utterance, with the referents of all of the indexicals resolved.

So under these definitions, the pair of sentences in (18) have the same character, while the pair in (19) have the same content. Kaplan's 'contents' are essentially the same as Carnap's 'intensions'; they are functions from possible worlds (a.k.a. 'circumstances of evaluation') to extensions. For sentences, the content is a proposition, a function from possible worlds to truth values. The character of a sentence is something that, given a context of utterance, gives you a content; formally a function from contexts to contents. So in a nutshell, the Kaplanian picture is as follows:

- Character + Context of utterance ⇒ Content
- Content + Circumstance of evaluation ⇒ Extension

(We could have written 'Intension' in place of 'Content' here; they play indistinguishable roles for our purposes.)

The CONTEXT OF UTTERANCE determines who is speaking, to whom, when, where, and in what possible world.

$$c = \langle sp, ad, t, loc, w \rangle$$

In Kaplan's 'logic of indexicals', there are certain special indexical constants, whose semantics are defined as follows:

(20) a. $[\![\mathsf{i}]\!]^{M,g,c} = sp(c)$
b. $[\![\mathsf{u}]\!]^{M,g,c} = ad(c)$
c. $[\![\mathsf{now}]\!]^{M,g,c} = t(c)$
d. $[\![\mathsf{here}]\!]^{M,g,c} = loc(c)$

English expressions can then be mapped to these special indexical constants like so:

(21) a. *I* ⇝ i
b. *you* ⇝ u
c. *now* ⇝ now

d. *here* $\rightsquigarrow$ here

Again, the CONTENT of a sentence is the proposition that is expressed after the reference of all of the indexicals has been fixed by the context of utterance. Formally, fixing g and c, the content of ϕ can be defined as:[4]

$$\{M : [\![\phi]\!]^{M,g,c} = 1\}$$

And again, the CHARACTER of a sentence is that aspect of its meaning that is the same across different contexts of use. This notion can be formalized as a function from contexts of utterance to contents. Fixing g, the character is that f such that:

$$f(c) = \{M : [\![\phi]\!]^{M,g,c} = 1\}$$

Now, Kaplan actually argues for this view of indexicals against an alternative theory according to which indexicals are disguised definite descriptions. One might imagine the following alternative analysis:

(22) $I \rightsquigarrow \iota x.\,\mathsf{Speaker}(x)$

(23) *you* $\rightsquigarrow \iota x.\,\mathsf{Addressee}(x)$

On this view, there is no need to posit a separate context of utterance.

This alternative view fails to account for the fact that the following two sentences have very different meanings.

(24) a. If I were male, I would not be speaking right now.
b. If the person speaking were male, I would not be speaking right now.

[4] The way Kaplan (1989) really defines it is closer to:

$$\{w : [\![\phi]\!]^{M,g,c,w} = 1\}$$

where $M = \langle D, I, W, C\rangle$ is an intensional model. We will introduce intensional models in Chapter 13.

Spoken by a female person, the first sentence would seem untrue. But the second sentence would seem true. The situation with proper names is analogous to the case with indexicals:

(25) a. If Liz were male, Liz would not be speaking right now.
b. If the person speaking were male, Liz would not be speaking right now.

The first sentence seems false, but the second sentence seems true. In any case, their meanings are very different.

Similarly, Ed's wish in the following two sentences is satisfied under very different circumstances:

(26) a. Ed wishes that I were male.
b. Ed wishes that the person speaking were male.

And again, the variant with the proper name patterns with the one with the indexical first person pronoun:

(27) Ed wishes that Liz were male.

According to Kaplan, indexicals, like proper names, are DIRECTLY REFERENTIAL: they refer to the same individual in every possible world. Unlike definite descriptions like *the speaker*, they do not look in a world to see who is the speaker there and then refer to that person. They directly pick out an element of the context of utterance. Definite descriptions like *the speaker*, in contrast, may refer to different individuals in different worlds. Although indexicals may be said to have descriptive content, it is part of their *character*, not their *content*.

Kaplan's conclusion is that we need to add CONTEXT OF UTTERANCE as a parameter according to which we determine the semantic value of linguistic expressions:

$$[\![\alpha]\!]^{M,g,c} = \ldots$$

We will adopt this context-of-utterance parameter in our treat-

ment of tense, which as Reichenbach (1947) points out, is a "particularly important" type of indexicality.

12.4 Tense

We are now ready to begin formalizing a theory of tense, building on the ideas of Reichenbach, Klein, and Kaplan. One of the questions that we must take a stand on is how to conceive of the reference time: Is it existentially quantified or free? In other words, does the past tense say 'there was some time R in the past such that...' or does it pick out a salient time R from the context?

In order to give a bit of historical context for early work on tense, let us begin by presenting Arthur Prior's tense logic, which is essentially an 'existential' theory of the past. We will then discuss its shortcomings, in comparison to an 'anaphoric' theory of the past.

12.4.1 Priorean tense logic

In Arthur Prior's TENSE LOGIC, a formula can vary in its truth value across time. Thus *Susan is asleep* might be true at time t, but false at time t'. A future sentence like *Susan will be asleep* can then be said to be true at time t if there is a time t' later than t at which Susan is asleep.

To achieve this, we will add to our models so that they consist not only of a domain of individuals D and an interpretation function I but also a set of times T and a linear ordering relation $<$ among the times. A TEMPORAL MODEL for a language L is then a quadruple

$$\langle D, I, T, < \rangle$$

such that D is a set of individuals, T is a set of times, $<$ is the 'earlier than' relation among the times, and I is an interpretation function which maps the non-logical constants to appropriate denotations at the various times. The function I will thus take two arguments:

a constant, and a time. For example, suppose we have a model in which the domain $D = \{a, b, c\}$, the set of times T is $\{t_1, t_2, t_3\}$, and we have two individual constants john and mary, and one predicate constant Happy. The interpretation function I might then be defined as follows:

$I(t_1, \textsf{john}) = b$	$I(t_2, \textsf{john}) = b$	$I(t_3, \textsf{john}) = b$
$I(t_1, \textsf{mary}) = a$	$I(t_2, \textsf{mary}) = a$	$I(t_3, \textsf{mary}) = a$
$I(t_1, \textsf{Happy}) = \{a, b, c\}$	$I(t_2, \textsf{Happy}) = \{a, b\}$	$I(t_3, \textsf{Happy}) = \{c\}$

So, throughout time, the name john always denotes the same individual, namely b, and so does the name mary. But who is happy changes. At first, everyone is happy, then c becomes unhappy, but c has the last laugh in the end.

Truth will be relative not only to a model and an assignment function, but also to a time, so we will have expressions like:

$$[\![\textsf{Happy(mary)}]\!]^{M,g,t_1} = 1$$
$$[\![\textsf{Happy(mary)}]\!]^{M,g,t_3} = 0$$

Both of these meta-language statements happen to be true according to the way we have set things up.

This framework allows for the definition of future and past operators. To the syntax of the language, we add the following rules:

- If ϕ is a formula, then $\mathbf{F}\phi$ is a formula.
- If ϕ is a formula, then $\mathbf{P}\phi$ is a formula.

($\mathbf{F}\phi$ can be read: 'it will be the case that ϕ', or 'future ϕ'; $\mathbf{P}\phi$ can be read: 'it was the case that ϕ', or 'past ϕ'.)

These kinds of statements can be given truth values relative to a particular time that depend on what value ϕ takes on at times preceding or following the evaluation time, respectively:

- $[\![\mathbf{F}\phi]\!]^{M,g,t} = 1$ iff $[\![\phi]\!]^{M,g,t'} = 1$ for some t' such that $t < t'$.

- $[\![\mathbf{P}\phi]\!]^{M,g,t} = 1$ iff $[\![\phi]\!]^{M,g,t'} = 1$ for some t' such that $t' < t$.

The way we have set things up, these formulas can be iterated ad infinitum, letting us model statements like 'Susan will have seen the report', which can take the form of $\mathbf{FP}\phi$ or 'A child was born that would become the ruler of the world' (Kamp, 1971), which might be modeled using a future operator in the scope of a past operator.

But let us not get too married to this system, because it suffers from a number of difficulties as a theory of tense. We turn to these next.

12.4.2 Shortcomings of the Priorean theory of tense

12.4.2.1 Partee's example

In Prior's tense logic, as we have just discussed, there is an operator **P** (for 'past') whose semantics is defined such that

$$\mathbf{P}\phi$$

is true at time t if *there is some time* t' prior to t such that ϕ is true at t'. For example, 'John sneezed' would be true at t if there is some time t' prior to t such that John sneezed at t'. This amounts to an *existential* theory of the past tense.

But consider a context 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:

(28) Oh no! I didn't turn off the stove!

The existential theory of the past tense does not make correct predictions about this case, as Partee (1973) famously pointed out. 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 (28) is saying something *about a particular time*.

Partee (1973) notes a number of 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 (28) is similar to a free pronoun, anaphorically referring back to to a time that has previously been introduced into the discourse.

12.4.2.2 Interactions between tense and aspect

Another shortcoming of Prior's theory of tense is that it has nothing to say about the interaction between tense and aspect. For example, both of the following sentences are in the past tense, but one implies that the event is complete, and the other allows for the possibility that the event is continuing:

(29) (When I was in the room,) Dave ate the cookie. (perfective)

(30) (When I was in the room,) Dave was eating the cookie. (imperfective)

The example in (29), which is in the simple past, has PERFECTIVE ASPECT. With perfective aspect, the past tense implies that the event in question has been completed. The past progressive example in (30) has IMPERFECTIVE ASPECT, which does not have the same implication; the event *might* still be going on. As pointed out by Klein (1994), this shows that it is not always the case that a past tense sentence means that the event described is in the past.

12.5 A formal theory of tense

12.5.1 Anaphoric theory of the past

Let us now present a theory of the past on which it refers to a salient past time, as Partee advocates. We will incorporate the ideas of Reichenbach, Klein and Kaplan in our theory as well.

As in Priorean tense logic, a model for our formal language specifies a set of times T, along with an ordering relation among the times $<$ as well as a containment relation among the times $\subseteq$. We extend the models for intensional logic that we had before, so a model M will have the following structure:

$$M = \langle D, I, W, T, <, \subseteq \rangle$$

where

- D is the domain of individuals D
- I is an interpretation function assigning semantic values to each of the non-logical constants in the language
- W is a set of worlds
- T is a set of times
- $<$ is a precedence relation among times
- $\subseteq$ is a containment relation among times

What constitutes the utterance time depends on the context of utterance, which means that tense morphology is indexical. Therefore, to model tense we will use an extension of Kaplan's system, where the semantic value of an expression is determined relative to a model M, an assignment function g, a world w, and a context c.

$$[\![\alpha]\!]^{M,g,w,c}$$

The 'utterance time' is the time determined by c, which we call $t(c)$.

Our formal language will allow expressions that refer to times. We will use i as the type designator for times, so expressions that refer to times will be of type i. We will allow an infinite set of variables of type i, so for example

$$v_{3,i}$$

is a variable of type i with index 3. We use

$$t_n$$

as an abbreviation for

$$v_{n,i}.$$

We will also use the symbols $<$ and $\subseteq$ in our logical language, and they will be interpreted as $<$ and $\subseteq$ in the model:

- Syntax: If α and β are expressions of type i, then,

  $$\alpha < \beta$$

  is a formula.

- Semantics:

  $$[\![\alpha < \beta]\!]^{M,g,w,c} = \begin{cases} 1 & \text{if } [\![\alpha]\!]^{M,g,w,c} < [\![\beta]\!]^{M,g,w,c} \\ 0 & \text{otherwise} \end{cases}$$

  (where $<$ is determined by M)

Similarly:

- Syntax: If α and β are expressions of type i, then, $\alpha \subseteq \beta$ is a formula.

- Semantics:

  $$[\![\alpha \subseteq \beta]\!]^{M,g,w,c} = \begin{cases} 1 & \text{if } [\![\alpha]\!]^{M,g,w,c} \subseteq [\![\beta]\!]^{M,g,w,c} \\ 0 & \text{otherwise} \end{cases}$$

  (where $\subseteq$ is determined by M)

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.

With these tools in hand, let us outline a simple theory of tense. The basic idea is that the past tense denotes a 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.

(31) $\text{PAST}_n \rightsquigarrow t_n$ (first version)

But there is an additional constraint. The past tense further requires that t_n precedes the time of utterance, while the present tense requires that t_n is identical to the time of utterance. We use the constant now to denote the time of utterance:

(32) $[\![\mathsf{now}]\!]^{M,g,c,w} = t(c)$

As Heim (1994) discusses, the denotation of the past tense should be undefined unless '$t_n < \mathsf{now}$' holds, because this constraint is more like a presupposition than an entailment. (If it were otherwise, 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.) As long as that constraint holds, the past tense should be the value of the assignment function for t_n. To get this result using the formal tools at our disposal, we can use an ι-expression, as follows:[5]

(33) $\text{PAST}_n \rightsquigarrow \iota t\,.\,[\,t = t_n \wedge t_n < \mathsf{now}\,]$

This expression will constrain both the assignment function g and the context of utterance c. The past tense will only have a defined value relative to assignment function g and context of utterance c

[5]This presupposition is analogous to the presupposition on gender features on pronouns: *he* presupposes that the reference is male. That presupposition can be captured by mapping he_i to the expression $\iota x\,.\,[\,x = v_i \wedge \mathsf{male}(v_i)\,]$.

when $g(t_n)$ precedes the time of utterance $t(c)$.

For now, the present tense will be analyzed simply as:

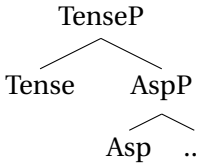
(34) PRESENT $\rightsquigarrow$ now

although there is evidence that the present tense behaves somewhat differently from the word *now* (Kamp, 1971). Compare:

(35) a. Someday Susan will marry a man she loves.
b. Someday Susan will marry a man she loves **now**.

These two sentences mean something different; the former describes a man she will love in the future; the latter describes a man she loves now. This contrast can be captured using Kratzer's (1998) notion of 'zero tense'. A 'zero tense' for Kratzer is an indexed time variable with no presuppositions (hence the name 'zero'), which must be bound by a local antecedent.[6] We will maintain the simple theory of the present tense in (34) for the time being, though.

Kratzer (1998) proposes that the syntax of verb phrases is layered so that an aspectual phrase, where the perfective/imperfective distinction is represented, dominates the VP, and a tense phrase, where the past/present distinction is represented, in turn dominates the aspectual phrase:



The node that AspP dominates is taken to denote a property of times, type $\langle i, t \rangle$. The AspP node imposes further constraints on

[6] Kratzer (1998) analogizes zero tenses to the phenomenon observed in sentences like *Only I did my homework*, where the first person possessive pronoun *my* seems to be interpretable without its first person feature, 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.'

this property of times, and this property is predicated of the time denoted by the Tense node.

Verbal predicates will take time arguments. For example:

(36) $dance \rightsquigarrow \lambda x.\lambda t.\mathsf{Dance}(t,x)$

This expression will thus denote a function from individuals to functions from times to truth values. So assuming that *Ann* $\rightsquigarrow$ a, *Ann dance* will be interpreted as:

$$\lambda t.\mathsf{Dance}(t,\mathsf{a})$$

In an event-semantic framework, it is assumed that verbs like *dance* denote properties of events. These events are assumed to have 'temporal traces' – the interval of time during which they occur. The temporal trace of an event e is usually denoted $\tau(e)$. In such a framework, the temporal argument of a verb would be introduced separately, yielding a predicate of times like:

$$\lambda t.\exists e[\mathsf{Dancing}(e) \wedge \mathsf{Agent}(e,\mathsf{a}) \wedge \tau(e) = t]$$

for *Ann dance*. This could then combine with aspectual and tense morphology in the same way, which we are about to see.

An Asp node will dominate either PERF for 'perfective' or IMP for 'imperfective'. Perfective aspect has the following interpretation:

(37) PERF $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\exists t'.t' \subseteq t \wedge P(t')$
'Takes a predicate of times P, and returns a predicate of times that is true of a time t if t' is contained in a time t at which P is true.'

(38) IMP $\rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\exists t'.t \subseteq t' \wedge P(t')$
'Takes a predicate of times P, and returns a predicate of times that is true of a time t if t is contained in a time t' at which P is true.'

The event time in these formulas corresponds to t', because that is the time of which P is predicated; see $P(t')$ in the formula. The

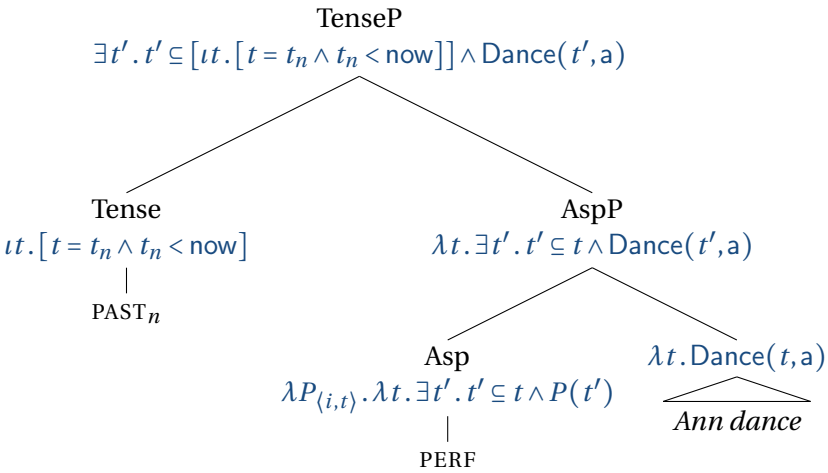


Figure 12.1: Derivation for *Ann danced*.

topic time is t. The difference between perfective and imperfective aspect is captured by the underlined portion in the informal glosses: With perfective aspect, the requirement is that the topic time t contains the event time t'. With imperfective aspect, the requirement is the other way around: that the topic time t is contained in the event time t'.

Thus for an example like *Ann danced*, we have the derivation in Figure 12.1: The top node introduces the presupposition $t_n <$ now. This presupposition can be extracted from the formula, yielding the simpler formula:

$$\exists t'.\, t' \subseteq t_n \wedge \partial(t_n < \text{now}) \wedge \text{Dance}(t', \text{a})$$

In this formula (and the equivalent one at the top of the tree in Figure 12.1), t_n 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.

Exercise 1. Write out how you would read the formula at the top of Figure 12.1 aloud.

Exercise 2. Explain how this treatment explains the 'completion inference' of the past perfect, i.e., the fact that *Ann danced* implies that there is a dancing event carried out by Ann that has reached completion.

Exercise 3. Compute a tree for *Ann was dancing*.

12.6 Future (in English)

It is natural to suppose that the English verb *will* denotes a time located after utterance time. However, many authors claim that the future is not a true tense. Evidence for this idea comes from the fact that there seems to be a past-tense version of *will*, namely *would*, seen in:

(39) (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.

The other element we represent as WOLL and treat on a par with the aspectual markers PERF and IMP.

(40) $\text{WOLL} \rightsquigarrow \lambda P_{\langle i,t\rangle}.\lambda t.\left[\exists t'.\, t < t' \wedge P(t')\right]$

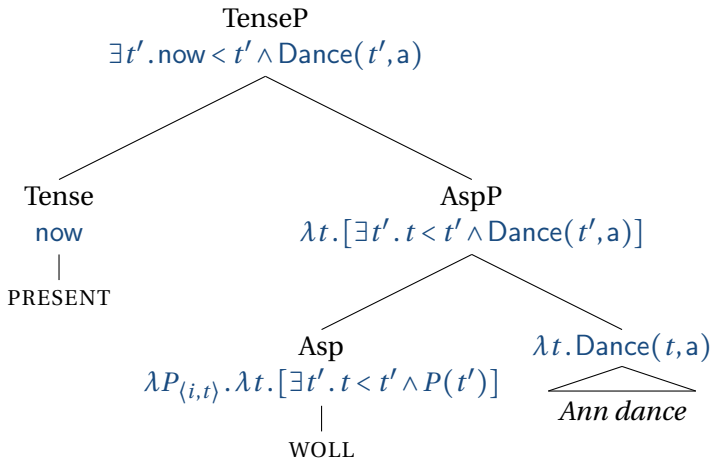


Figure 12.2: Derivation for *Ann will dance*.

This can combine with both present and past morphology. The verb *will* is the combination of present tense with WOLL; the verb *would* combines past and WOLL. A sentence like *Ann will dance* will have the representation in Figure 12.2, then.

On this view, English *will* does not occupy the same syntactic position as present or past morphology, nor does it describe the relationship between topic time and utterance time. Rather, it combines WOLL which is more like an aspectual marker with one of the two tenses, present or past. Thus the 'future' is not a tense in English. A similar claim has been made for St'át'imcets (Salish) by Matthewson (2006).

Exercise 4. Give a tree for *Ann would dance*.

12.6.1 Sequence of tense

One phenomenon that we have not covered is so-called 'sequence of tense' phenomena. Examples include the following:

(41) 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)

(42) John said he would buy a fish that **was** still alive. (Ogihara, 1989)

(43) 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 | Modality

13.1 Introduction

Note: This chapter is still in a preliminary stage.

13.2 Opacity

The following might seem like a well-founded principle to adopt in everyday reasoning:

(1) **Substitutability of coextensionals**
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.

For example, the following argument appears valid:

(2) a. Sir Walter Scott gave a public lecture.
b. Sir Walter Scott is the author of *Waverley*.
c. ∴ The author of *Waverley* gave a public lecture.

The conclusion seems to be licensed by the fact that Sir Walter Scott is the author of the novel *Waverley* – they are the same person. Therefore any property that the one has, the other has as well.

But there are examples where the principle of substitutability of coextensionals does not hold.

For instance, the author of *Waverley* happens to be Sir Walter Scott. (This is a famous set of examples by Russell. The novel was so popular in 19th century Britain that subsequent novels by the same author were sometimes billed simply as "By the author of *Waverley*".) Perhaps you did not know this already. Then someone could tell you (3) and you would learn something.

(3) Scott is the author of *Waverley*.

On the other hand, if someone were to tell you (4),

(4) Scott is Scott.

you would not learn anything new. (Throughout this chapter we set aside cases in which there are two or more people called Scott.)

It also seems much easier to agree that

(5) Scott might not have been the author of *Waverley*.

than that

(6) Scott might not have been Scott.

It is not hard to justify agreeing to (5). There was nothing inevitable about Scott's career. If he had decided to become, say a patent officer, he might well not have written any novels. There is a natural reading of (5), perhaps its most prominent one, which seems true for this reason. By contrast, even if Scott had become a patent officer, he would still have been Scott. Indeed, it seems hard to imagine what it would even take for (6) to be true.

Concomitantly, the following two sentences differ in their truth value, despite differing only in the substitution of one term for another that co-refers with it:

(7) a. Necessarily, Scott is the author of *Waverley*.
b. Necessarily, Scott is Scott.

The reason for the non-substitutability of coextensionals in this

case is that *necessarily* depends on the *proposition expressed* by the sentence it operates on, and not just on its *truth value*. This property renders *necessarily* unlike, say, negation, which depends only on the truth value; negation, as you may recall, is a 'truth-functional' connective. Whether or not a proposition *necessarily* holds depends on its truth value in *every* world, not just the world under consideration. In other words, *necesarily* depends on the INTENSION of the sentence it combines with, and not just its EXTENSION. The extension of an expression is its semantic value at a particular world (so, for formulas, the extension is a truth value), while the intension is a function from possible worlds to the extensions they have at those worlds. Expressions that depend on the intensions of the phrases they combine with, and not just their extensions, are called INTENSIONAL.

Verbs expressing attitudes towards propositions (PROPOSITIONAL ATTITUDE VERBS) such as *believe*, *know* and *want* are also intensional, as they express attitudes towards propositions, and not just truth values. We find violations of substitutability of coextensionals here as well. For example, *Mary believes that Scott is Scott* does not imply that *Mary believes that Scott is the author of* Waverley .

Some verbs of propositional attitude do not imply the truth of their complement. For example, (8a) does not imply (8b):

(8) a. Mary believes that Fred is in Paris.
b. Fred is in Paris.

Similarly, (9a) does not imply (9b).

(9) a. Mary believes that a unicorn is eating her parsnips.
b. A unicorn is eating Mary's parsnips.

In fact, (9a) does not even commit the speaker to the existence of unicorns, although it does imply that Mary believes that there are unicorns. In other words, (9a) lacks EXISTENTIAL IMPORT with respect to the indefinite *a unicorn*.

Propositional attitudes can be embedded in transitive verbs

that take a noun phrase direct object, and in such cases we observe the same effect. Thus, while (10a) implies that there is at least one unicorn, (10b) need not do so:

(10) a. Andrea sees a unicorn.
b. Andrea wants a unicorn.

To paraphrase Quine (1956), example (10b) can be interpreted to mean that Andrea merely seeks relief from unicorn-less-ness, not that there is a particular unicorn that Andrea wants; no particular unicorn need even exist for the sentence to be true. Thus a representation of the following kind would not do:

$$\exists x.[\mathsf{Unicorn}(x) \wedge \mathsf{Wants}(\mathsf{a}, x)]$$

because this implies that there are unicorns. How, then, should the meaning of a sentence like (10b) be represented? We will need to augment our representation language with tools for talking about other possible worlds in order to capture the meaning of these intensional expressions.

Before we embark on this task, however, we must observe that (10b) is ambiguous; it could be interpreted to mean that there is a particular unicorn that Andrea wants. The two readings involved here are called DE RE ('of the object') and DE DICTO ('of the word'). On the *de re* reading, Andrea has a belief about a particular unicorn: Regarding *that* unicorn, she wants it. The *de dicto* reading is the one on which she merely seeks relief from unicorn-less-ness. In the latter case, the desire is not about a particular individual, rather it is about the category, unicorns; she wants that category to be instantiated in her possession. In this sense, the desire is about the 'word', naming the category.

Quine (1956), two whom the *de dicto* vs. *de re* distinction is usually attributed, illustrated the *de dicto* / *de re* ambiguity with the following example:

(11) Ralph believes that someone is a spy.

On the *de re* reading, Ralph has a belief about a particular object/individual: There is someone about whom Ralph believes that they are 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 category, spies. Ralph believes that the category is instantiated. The *de dicto* interpretation does not commit the speaker to the existence of spies; only Ralph.

For another example:

(12) John believes that a Republican will win.

On one interpretation, there is a specific Republican who John believes will win. John may not even *know* that the person in question is a Republican. This is the *de re* interpretation. On the *de dicto* interpretation, there is no specific Republican that John believes will win; he just believes that whoever wins will be of that party.

Borrowing Quine's metaphor, verbs like *believe* 'seal off' the complement clause. As a consequence, the existential import of the complement clause is not inherited by the sentence as a whole. In this sense, the verbs are OPAQUE. (The opposite of opaque is TRANSPARENT.) When an indefinite is interpreted within the scope of an opaque verb (i.e., with a *de dicto* reading), the sentence as a whole does not carry the existential import that the indefinite would normally produce.

Exercise 1. Give three more examples of opaque verbs and three more examples of transparent verbs. Be warned that there are some borderline cases, where intuitions are murky; do your best to stick to clear cases.

This kind of ambiguity occurs not only in philosophy texts, but also 'in the wild', or at least, in legal statutes. Anderson (2014) describes the following case.

> In the fall of 2001, the accounting firm Arthur Andersen directed a large scale destruction of documents regarding its client Enron. Expecting a federal subpoena of records as a wave of accounting scandals unfolded, the firm urged its employees to begin shredding papers in October, shortly before the SEC began an official investigation into Enron. The shredding ceased abruptly on November 9th, immediately on the heels of the SEC's subpoena. In 2005, the Supreme Court reversed Arthur Andersen's conviction for "knowingly . . . corruptly persuad[ing] another . . . with intent to . . . induce any person to . . . withhold a record, document, or other object, from an official proceeding." The conviction was defective in part because the jury instructions did not make clear that the defendant's actions had to be connected to a particular official proceeding that it had in mind, which in this case had not been initiated at the time of the shredding. The ruling followed a line of obstruction of justice decisions dating back to the nineteenth century in holding that, if in its frenzy of paper shredding the defendant firm was not specific about the particular official proceeding to be obstructed, the statute could not have been violated.

On the *de re* interpretation (for the both the document and the proceeding from it) of the statute, it is violated when there is a particular document from a particular official proceeding which the perpetrator intends to withhold. On a *de dicto* interpretation, it is violated when the intent is such that there is an official proceeding from which documents are withheld. It is fair to say that Andersen would be guilty on a *de dicto* interpretation, and were acquitted on the basis of a *de re* interpretation.

Exercise 2. Consider the following case from Anderson (2014):

> In 1869, an English court considered the case of Whiteley v. Chappell, in which a man who had voted in the name of his deceased neighbor was prosecuted for having fraudulently impersonated a "person entitled to vote." The court acquitted him, albeit reluctantly. There had been voter fraud by impersonation, certainly. But the court fixated on the object of the impersonation and concluded that because a dead person could not vote, the defendant had not impersonated a "person entitled to vote." The court attributed the mismatch between this result and the evident purpose of the statute to an oversight of the drafters: "The legislature has not used words wide enough to make the personation of a dead man an offence."

How would you characterize the *de re* and *de dicto* interpretations, respectively, in this case? Which interpretation does the court appear to have taken? Is there an interpretation on which the man is guilty? Explain why or why not.

A good theory of propositional attitude verbs should be able to account for the *de dicto* vs. *de re* ambiguity. In order to build up the theoretical machinery necessary in order to do this, we will start with modal logic, which contains the intensional operators *necessarily* and *possibly*. We then present Montague's Intensional Logic, which builds on modal logic and provides a mechanism for compositional interpretation of sentences involving intensional operators.

13.3 Modal Logic

13.3.1 Alethic logic

As discussed above, (13a) (on its most natural reading) expresses a truth that is not a necessary truth; things could have been otherwise, for example if Scott had become a patent officer instead of an author. By contrast, a statement like (13b) has a different status: it is necessarily true.

(13) a. Scott is the author of *Waverley*.
b. Scott is Scott.

In other words, the statement in (13a) is CONTINGENTLY TRUE, while the statement in (13b) is NECESSARILY TRUE.

We can also divide false statements into those that are necessarily false and those that are contingently false in an analogous manner. For example, both (14a) and (14b) are false, but the former is contingently false and the latter is necessarily false.

(14) a. Scott is the author of *Frankenstein*.
b. Scott is not Scott.

Exercise 3. Give another example of a contingently false statement and one example of a necessarily false statement.

A logical system representing concepts like *it is necessary that* and *it is possible that* is called an ALETHIC LOGIC or MODAL LOGIC. The term 'modal logic' is somewhat more common and frequent, but it also has a broader usage, sometimes also applying to tense logics of the Priorian kind. The operator 'it is necessary that' is standardly represented as a box, $\Box$, and 'it is possible that' is represented as a diamond, $\Diamond$. Thus in alethic logic the syntax rules are extended with the following:

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

From the perspective of the system that we have developed so far, it may seem natural to define the semantics of $\Box\phi$ by saying that the formula is true if ϕ is true in every first-order model. This is how Rudolph Carnap defined it. A slight variant on this view is due to Saul Kripke, who contributed a new notion of model. A model in Kripke's framework contained a *set* of first-order models, each representing a different possible state of affairs, or a POSSIBLE WORLD. In this way, Kripke formalized an idea from Leibniz that a necessary truth is one that is true in all possible worlds. These so-called KRIPKE MODELS had a flexibility that was absent from Carnap's system.

Now, a first-order model consists of a domain and an interpretation function. So in principle, the possible worlds in a Kripke model might have different domains. But we will assume for simplicity (and not without good philosophical reason) that there is a single domain of individuals that is shared across all possible worlds. A model for modal logic will therefore consist of a set of possible worlds W, in addition to a domain of individuals D and an interpretation function I. Unlike in tense logic, the worlds are not ordered. Thus a model will be a triple:

$$\langle D, W, I \rangle$$

where D is a set of individuals, W is a set of worlds, and I is an interpretation function. Just as in tense logic, the interpretation function I will take two arguments: a non-logical constant, and, this time, a world. So if there are three worlds w_1, w_2, and w_3, and three individuals a, b and c, it might be the case that

$$I(w_1, \mathsf{Happy}) = \{a, b, c\}$$

but

$$I(w_3, \mathsf{Happy}) = \{c\}$$

Truth of a formula will in general be relative to a model M, an assignment function g, and a possible world w. So assuming that john maps to a in every possible world, we have:

$$[\![\mathsf{Happy(john)}]\!]^{M,g,w_1} = 1$$

but

$$[\![\mathsf{Happy(john)}]\!]^{M,g,w_3} = 0$$

Note that there is controversy as to how possible worlds should be conceived of. On David Lewis's view they are physical objects, such as the universe we actually live in. On another view, which can be attributed to Ludwig Wittgenstein, they are merely ways the world can be.[1] The formalization here does not depend on a particular conception of possible worlds, but the authors tend to prefer Wittgenstein's perspective.

The semantics of the modal operators can be defined syncategorematically as follows:[2]

- $[\![\Box\phi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w'} = 1$ for all w'
- $[\![\Diamond\phi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w'} = 1$ for some w'

It turns out that, using these definitions, certain intuitively valid sentences are indeed valid, for example:

- $\Box\phi \leftrightarrow \neg \Diamond \neg\phi$
  'It is necessarily the case that phi if and only if it is not possible that not phi'
- $\Box\phi \rightarrow \phi$
  'Necessarily phi implies phi'

[1] See `https://plato.stanford.edu/entries/possible-worlds/`.

[2] This is one of many possibilities; this simplest system is known as S5. See Hughes & Cresswell (1968) for a fuller presentation. For the notion SYNCATEGOREMATIC, see Section 7.3.

- $\phi \rightarrow \Diamond \phi$
  'If phi, then possibly phi'

The first statement implies that $\Diamond$ is the DUAL of $\Box$. (In the same way, $\exists$ is the dual of $\forall$, since $\forall x . \phi$ is equivalent to $\neg \exists x . \neg \phi$.) In fact, sometimes in modal logic a statement of the semantics of $\Diamond$ is left out, and the $\Diamond$ is defined as a syntactic abbreviation of $\neg \Box \neg$.

Given our semantics for the quantifiers from previous chapters, the following formulas are also valid:

- $\forall x \Box \phi \rightarrow \Box \forall x \phi$
- $\exists x \Diamond \phi \rightarrow \Diamond \exists x \phi$

The first of these is known as the BARCAN FORMULA; the second is actually equivalent. But as Dowty et al. (1981, 129) explain:

> [I]t is somewhat controversial whether [these two statements] should be formally valid. It has been suggested that $\forall x \Box \phi$ ought to mean, "every individual x that *actually* exists is necessarily such that ϕ, whereas $\Box \forall x \phi$ ought to mean "in any possible world, anything that exists in that possible word is such that ϕ." Similarly, $\exists x \phi$ ought to mean that "some individual x that actually exists is in some world such that ϕ", whereas $\Diamond \exists x \phi$ should mean that "in some world it is the case that some individual which exists in that world is such that ϕ." To make these pairs of formulas semantically distinct would require a model theory in which each possible world has its own domain of individuals over which quantifiers range (though the domains would, in general, overlap partially). In this way, there could be "possible individuals" that are not actual individuals, and perhaps actual individuals that do not "exist" in some other possible worlds. The question whether there are such individuals has, not surprisingly, been the subject of considerable philosophical debate. It

> is possible to construct a satisfactory model theory on this approach (and in fact Kripke's early treatment in Kripke 1963 adopted it), but it is technically more complicated than the approach we have adopted here, and it was not adopted by Montague (for discussion see Hughes & Cresswell 1968, pp. 170–184).

Note that treating possible worlds as first-order models, as Carnap did, naturally suggests that different possible worlds may well be associated with different domains of individuals. This not only makes things more complicated, it also raises issues related to how one might recognize a given individual as 'the same' individual across worlds, which of course is important for capturing the semantics of sentences like *I could have been a millionaire*. Lewis (1968) advocates an extreme version of the differing-domains view, on which no two worlds share individuals. Rather, individuals are identified across worlds through a COUNTERPART RELATION. In a system where there is a fixed domain for all possible worlds, this problem does not arise.

However, there are examples that seem to suggest that the verb *exist* denotes a contingent property (examples from Coppock & Beaver 2015):

(15) My university email account no longer exists.

(16) If that existed, then I would have heard of it!

A famous example discussed by Russell (1905) is:

(17) The golden mountain does not exist.

This sentence is felt to be true; but in that case, what does *the golden mountain* refer to? One way of capturing these facts in a fixed-domain framework is to introduce an existence predicate exists, understood to be true of an individual at a world if that individual really exists at that world. Thus we can distinguish between two kinds of 'existence': BROAD EXISTENCE, which holds of

everything a quantifier can range over, that is, everything in the domain of individuals, and NARROW EXISTENCE, which is a contingent property of individuals, holding at some worlds but not others. The verb *exist* can be taken to denote the narrow, contingent kind of existence, captured by the existence predicate.

13.4 Intensional Logic

13.4.1 Introducing Intensional Logic

In the previous section, we defined the semantics of $\Box$ and $\Diamond$ syncategorematically, rather than giving $\Box$ a meaning of its own. It is common to do this with negation as well. But with negation, unlike necessity, it is possible to give the symbol a meaning of its own, one that is a function of the truth value of its complement. The semantic value of the $\neg$ symbol can be defined as a function that returns 0 if it receives 1 as input, or 1 if it receives 0 as input. The same is not the case for $\Box$, because the truth of a $\Box$ statement depends not on the truth value of its complement at a particular world. We saw this above with the following examples:

(18) a. Necessarily, Scott is the author of *Waverley*.
b. Necessarily, Scott is Scott.

Indeed, the truth of a necessity statement depends on the whole range of truth values that the inner formula takes on across all worlds. So if $\Box$ denotes a function, it does not take as input a truth value. Rather, it must take as input a specification of all of the truth values that the sentence takes on, across all worlds. In other words, the input to the function that $\Box$ denotes must be a proposition. In this section, we will develop tools that make it possible to give $\Box$ a denotation of its own, and feed the right kind of object to it as an argument.

The technique we will use (due to Carnap) is to associate with each expression both an INTENSION and an EXTENSION. The in-

tension is a function from possible worlds to the denotation of the expression at that world. The denotation of an expression at a world is called the extension (of the expression at that world).

- A name (type e), which denotes an individual, has an intension that is a function from possible worlds to individuals. A function from possible worlds to individuals is called an INDIVIDUAL CONCEPT.
- A unary predicate (type $\langle e, t \rangle$), which denotes a set of individuals (or characteristic function thereof), has a function from possible worlds to (characteristic functions of) sets of individuals as its intension. Such a function is called a PROPERTY.
- A formula (type t), which denotes a truth value, has as its intension a function from possible worlds to truth values. A function from possible worlds to truth values is called a PROPOSITION.

The extension of an expression α at world w (with respect to model M and assignment function g) is denoted by $[\![\alpha]\!]^{M,g,w}$. The intension of an expression α is that function f such that $f(w) = [\![\alpha]\!]^{M,g,w}$. That function is sometimes denoted as follows:

$$[\![\alpha]\!]^{M,g}_{¢}$$

with the cent sign ¢ as a subscript on the denotation brackets, and no world variable superscript.

Note that it is not possible to figure out the intension from the extension at a particular world. In order to get the intension, you need to know the extension at *every* possible world. So there is no function from extensions to intensions. Note also that every expression in the language gets an extension, even variables. But since the denotation of a variable is always determined by an assignment function, its intension relative to g will be a function that yields the same value for every possible world given as input.

Now let us return to the problem of giving a compositional, non-syncategorematic semantics for necessity and belief. Recall that if the $\Box$ operator denotes any function, it denotes one whose input is a proposition, rather than a truth value. The relevant proposition is of course the intension of the formula with which it combines. The strategy that Montague followed in order to do so was to introduce a device that forms from any expression α a new expression denoting the intension of α. The device is called the 'hat operator', and it looks like this:

$$\hat{}\alpha$$

Relative to any given world, this expression has as its *extension* the *intension* of α. For example, the formula

$$\mathsf{Happy(m)}$$

has either 1 or 0 as its extension in every world. In w_1, the extension of this formula might be 1; in w_2, the extension might be 0; in w_3, the extension might be 1. The intension is a function from worlds to truth values. But the expression:

$$\hat{}\mathsf{Happy(m)}$$

has the intension of $\mathsf{Happy(m)}$ as its extension. (Put more simply: The extension of $\hat{}\mathsf{Happy(m)}$ is the intension of $\mathsf{Happy(m)}$.) We now therefore have a new class of expressions, which denote functions from possible worlds to other sorts of things. With the help of this 'hat' operator, a formula, which normally denotes a truth value, can be converted into an expression that denotes a proposition. This new expression is the right kind of input for an expression that denotes necessity or belief.

Before showing how that works, it will be convenient to define an addition to the type system that allows for the new kinds of expressions that are formed using this operator. Letting s stand for the type of possible worlds, we now have, for every type τ, a new type $\langle s, \tau \rangle$. The complete type system is now as follows:

- t is a type
- e is a type
- If σ and τ are types, then so is $\langle \sigma, \tau \rangle$
- If τ is any type, then $\langle s, \tau \rangle$ is a type.

Our syntax rules will be extended so that if α is an expression of type τ, then $\hat{}\alpha$ is an expression of type $\langle s, \tau \rangle$. Any expression of type $\langle s, \tau \rangle$ will denote a function from possible worlds to D_τ, where D_τ is the domain of entities denoted by expressions of type τ. The official semantic rule for $\hat{}$ is as follows:

- If α is an expression of type τ, then $[\![\hat{}\alpha]\!]^{M,g,w}$ is that function f with domain W such that for all $w \in W$: $f(w)$ is $[\![\alpha]\!]^{M,g,w}$.

The hat operator also has a partner, $\check{}$ which moves from intensions to extensions. If α is an expression of type $\langle s, \tau \rangle$, then $\check{}\alpha$ is an expression of type τ. Its semantics is defined as follows:

- If α is an expression of type $\langle s, \tau \rangle$, then $[\![\check{}\alpha]\!]^{M,g,w}$ is the result of applying the function $[\![\alpha]\!]^{M,g,w}$ to w.

With these tools in hand, let us now consider how we might get a handle on the *de dicto* / *de re* ambiguity and related puzzles. Let us introduce a constant bel, which relates a proposition (denoted by an expression of type $\langle s, t \rangle$) with an individual (denoted by an expression of type e). Given that *believe* combines first with its clausal complement and then with its subject, its type should then be

$$\langle \langle s, t \rangle, \langle e, t \rangle \rangle$$

The *de dicto* reading of a sentence like *John believes that a Republican will win* can then be represented as follows:

$$\mathsf{Bel}(\mathsf{john}, \hat{}\exists x[\mathsf{Repub}(x) \wedge \mathsf{Win}(x)])$$

The *de re* reading can be represented:

$$\exists x[\mathsf{repub}(x) \wedge \mathsf{bel}(\mathsf{john}, \hat{}[\mathsf{win}(x)])$$

In the latter formula, the existential quantifier and the predicate Repub occur outside the scope of the belief operator. So on the *de re* reading, John's belief does not have to do with the property of being a Republican; it's about the particular individual. Not so for the *de dicto* reading, on which the content of John's belief involves that property.

Let us consider one more example of a *de dicto* / *de re* ambiguity, this time involving the proper name *Miss America*. The following sentence can be understood in two ways:

(19) John believes that Miss America is bald.

On the *de re* interpretation, John believes of some particular individual that she is bald. As it happens, this individual is also Miss America, although John doesn't have to know that as far as the *de re* interpretation is concerned. On the *de dicto* interpretation, John believes that whoever is Miss America is bald. As far as this interpretation is concerned, John doesn't have any acquaintance at all with the individual who is Miss America, and doesn't have to know who she is. The two interpretations can be represented as follows:

(20) $[\lambda x.\mathsf{Bel}(\mathsf{john}, \hat{}\mathsf{Bald}(x))](\mathsf{m})$

(21) $\mathsf{Bel}(\mathsf{john}, \hat{}\mathsf{Bald}(\mathsf{m})]$

As the identity of Miss America varies from situation to situation, let us assume that this name is *not* a rigid designator, but rather a non-logical constant whose value can vary from world to world. Then, the first of the two formulas above captures the *de re* interpretation; the second captures the *de dicto* one. When m is in the scope of the Bel operator, its interpretation may vary from world to world, but when it is outside, it just denotes whoever Miss America is in the current world.

This example has an important consequence: lambda-conversion is not in general a valid principle anymore. When the lambda-bound variable is found in the scope of an intensional operator, lambda conversion can change the meaning. See Dowty et al. (1981, ch. 6, pp. 166-167) for a somewhat fuller discussion of this issue.

13.4.2 Formal fragment

Let us define a new logic, IL 'Intensional Logic', following Montague. The language is not exactly the same as Montague's Intensional Logic, but it is fundamentally similar in spirit.

13.4.2.1 Semantics

The types are defined recursively as follows:

- t is a type
- e is a type
- If σ and τ are types, then so is $\langle \sigma, \tau \rangle$
- If τ is any type, then $\langle s, \tau \rangle$ is a type.

The set of expressions of type τ, for any type τ, is defined recursively as follows:

1. **Basic Expressions**
   For each type τ,
   (a) the **non-logical constants** of type τ are the symbols of the form $c_{n,\tau}$ for each natural number n.
   (b) the **variables** of type τ are the symbols of the form $v_{n,\tau}$ for each natural number n.
2. **Predication**
   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.

3. **Equality**
   If α and β are terms, then $\alpha = \beta$ is 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 $\neg\phi$, $[\phi \wedge \psi]$, $[\phi \vee \psi]$, $[\phi \rightarrow \psi]$, and $[\phi \leftrightarrow \psi]$.

6. **Quantification**
   If ϕ is a formula and u is a variable of any type, then $[\forall u.\phi]$ and $[\exists u.\phi]$ are formulas.

7. **Lambda abstraction**
   If α is an expression of type τ and u is a variable of type σ then $[\lambda u.\alpha]$ is an expression of type $\langle\sigma,\tau\rangle$.

8. **Alethic modalities** (new!)
   If ϕ is a formula, then $\Box\phi$ and $\Diamond\phi$ are formulas.

9. **Intensionalization** (new!)
   if α is an expression of type τ, then $\hat{}\alpha$ is an expression of type $\langle s,\tau\rangle$.

10. **Extensionalization** (new!)
    If α is an expression of type $\langle s,\tau\rangle$, then $\check{}\alpha$ is an expression of type τ.

The semantic values of expressions in IL depend on a model, an assignment function, and a world. A model $M = \langle D, I, W\rangle$ is a triple consisting of the domain of individuals D, an interpretation function I which assigns semantic values to each of the non-logical constants in the language, and a set of worlds W.

Types are associated with domains. The domain of individuals $D_e = D$ is the set of individuals, the set of potential denotations for an expression of type e. The domain of truth values D_t contains just two elements: 1 'true' and 0 'false'. For any types a and b, $D_{\langle a,b\rangle}$ is the domain of functions from D_a to D_b. For every type a, I assigns an object in D_a to every non-logical constant of type a.

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 a a denotation from the set D_a.

The semantic value of an expression is defined as follows:

1. **Basic Expressions**

   (a) If α is a non-logical constant, then $[\![\alpha]\!]^{M,g,w} = I(w,\alpha)$.

   (b) If α is a variable, then $[\![\alpha]\!]^{M,g,w} = g(\alpha)$.

2. **Predication**
   If α is an expression of type $\langle a,b\rangle$, and β is an expression of type a, then $[\![\alpha(\beta)]\!] = [\![\alpha]\!]([\![\beta]\!])$.

3. **Equality**
   If α and β are terms, then $[\![\alpha = \beta]\!]^{M,g,w} = 1$ iff $[\![\alpha]\!]^{M,g,w} = [\![\beta]\!]^{M,g,w}$.

4. **Negation**
   If ϕ is a formula, then $[\![\neg\phi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w} = 0$.

5. **Binary Connectives**
   If ϕ and ψ are formulas, then:

   (a) $[\![\phi \wedge \psi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w} = 1$ and $[\![\psi]\!]^{M,g,w} = 1$.

   (b) $[\![\phi \vee \psi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w} = 1$ or $[\![\psi]\!]^{M,g,w} = 1$.

   (c) $[\![\phi \rightarrow \psi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w} = 0$ or $[\![\psi]\!]^{M,g,w} = 1$.

   (d) $[\![\phi \leftrightarrow \psi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w} = [\![\psi]\!]^{M,g,w}$.

6. **Quantification**

(a) If ϕ is a formula and v is a variable of type a then $[\![\forall v.\phi]\!]^{M,g,w} = 1$ iff for all $k \in D_a$:

$$[\![\phi]\!]^{M,g[v\mapsto k],w} = 1$$

(b) If ϕ is a formula and v is a variable of type a then $[\![\exists v.\phi]\!]^{M,g,w} = 1$ iff there is an individual $k \in D_a$ such that:

$$[\![\phi]\!]^{M,g[v\mapsto k],w} = 1.$$

7. **Lambda Abstraction**
If α is an expression of type a and u a variable of type b then $[\![\lambda u.\alpha]\!]^{M,g,w}$ is that function h from D_b into D_a such that for all objects k in D_b, $h(k) = [\![\alpha]\!]^{M,g[u\mapsto k],w}$.

8. **Alethic modalities** (new!)

(a) $[\![\Box\phi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w'} = 1$ for all w'

(b) $[\![\Diamond\phi]\!]^{M,g,w} = 1$ iff $[\![\phi]\!]^{M,g,w'} = 1$ for some w'

9. **Intensionalization** (new!)
If α is an expression of type τ, then $[\![\hat{}\alpha]\!]^{M,g,w}$ is that function f with domain W such that for all $w' \in W$: $f(w')$ is $[\![\alpha]\!]^{M,g,w'}$.

10. **Extensionalization** (new!)
If α is an expression of type $\langle s,\tau\rangle$, then $[\![\check{}\alpha]\!]^{M,g,w}$ is the result of applying the function $[\![\alpha]\!]^{M,g,w}$ to w.

Exercise 4. Formalize the *de dicto* and *de re* readings of the what the statute prohibits in the Andersen example on page 459, and describe in your own words what the truth conditions are under these two readings; in other words, describe what properties a model would have to have in order for the reading to be true.

Exercise 5. Is it possible to give a non-syncategorematic treatment of the hat operator ^? Explain why or why not.

13.5 Fregean sense and hyperintensionality

Frege's assessment of his puzzle about identity built on a distinction between *sense* and *reference*. For Frege, expressions such as "Scott" and "the author of *Waverley*" have the same referent, but they differ in their sense. Frege was not fully explicit about what a sense was, but described it as a 'mode of presentation'. A Carnapian intension is like a Fregean sense insofar as it provides a more fine-grained notion of meaning, but one might question whether it really captures what Frege had in mind. Perhaps Frege's notion of sense is even more fine-grained than the notion of intension.

Certainly, intensions in Carnap's sense are not sufficiently fine-grained to capture entailment relations among belief sentences. For example, the sentence $2 + 2 = 4$ is a mathematical truth, so it is true in every possible world. And there are many other mathematical truths that are true in exactly the same possible worlds (namely all of them), such as the fact that there are infinitely many prime numbers. . But from (22), it does not follow that (23) is true.

(22) Susan believes that $2 + 2 = 4$.

(23) Susan believes that there are infinitely many prime numbers.

This problem is not limited to tautologies; it also holds for pairs of contingent but logically equivalent propositions where the logical equivalence might be cognitively difficult to compute. For example, the law of contraposition is sometimes difficult for human beings to compute, so the (24) does not imply (25) (example from Muskens 2005):

(24) Susan believes that the cat is in if the dog is out.

(25) Susan believes that the dog is in if the cat is out.

Both of these cases exemplify the PROBLEM OF LOGICAL OMNISCIENCE: in general, people do not believe all of the logical consequences of their beliefs. Phenomena in which the substitution of one expression for another that has the same intension leads to a difference in truth value are called HYPERINTENSIONAL. Such cases clearly show that the analysis of belief given in the previous section is inadequate, and moreover that a more fine-grained notion of meaning is required. Recent perspectives on the problem are collected in a special volume of the journal *Synthese*; see Jespersen & Duží 2015 for an overview.

But the existence of hyperintensionaltiy does not negate the existence of the intensional 'layer' of meaning, as it were. Intensions are like the shadows of hyperintensions. And intensions are quite a bit more straightforward to deal with and more standard, at the time of writing. Therefore, in order to keep things manageable, we will continue to work 'at the intensional level' as it were, keeping in mind that any fully adequate theory ought to be hyperintensional.

13.6 Explicit quantification over worlds

The astute reader may have noticed that in the system described in §13.4, there were no expressions of type s. This is partly because Montague and his contemporaries believed that there were no expressions that made reference to possible worlds. That assumption has since been challenged, and for that reason among others, it is generally preferred nowadays to use a formal system in which there is explicit quantification and binding of possible worlds. On this view, rather than writing

$$\hat{}\mathsf{Bald}(\mathsf{m})$$

one writes rather:

$$\lambda w\,.\,\mathsf{Bald}_w(\mathsf{m})$$

where the subscript on w is meant to indicate that const denotes a function that takes a possible world as an argument, in addition to an individual.

The *de dicto* vs. *de re* ambiguity in

(26) John believes that a Republican will win.

would be captured as follows. The *de dicto* reading would be translated as follows:

$$\lambda w\,.\,\mathsf{Bel}_w(\mathsf{j}, \lambda w' \exists x[\mathsf{Republican}_{w'}(x) \wedge \mathsf{Win}_{w'}(x)])$$

The *de re* reading would be translated thus:

$$\lambda w\,.\,\exists x[\mathsf{Republican}_w(x) \wedge \mathsf{Bel}(\mathsf{j}, \lambda w'\,.\,\mathsf{Win}_{w'}(x))$$

Although the representation is a bit more cluttered, with world variables here and there, it does make explicit which world the various predicates hold in. In the *de dicto* formula, Republican is associated with w', the world that features in the content of John's belief, whereas in the *de re* reading, it is associated with w, the world of evaluation, i.e. the world in which John has the belief. This captures the fact that on the *de dicto* reading, there may be no particular Republican that John's attitude relates to; indeed, the sentence could be true even if Republicans did not exist.

A famous example of this sort of system is Gallin's (1975) Ty_2, so called because it has two types (s and e) other than t. The semantics literature still uses both styles, as each has its own advantages, although Gallin's style is gaining in popularity.

13.6.1 Modal auxiliaries

So far we have discussed two types of modal expressions: the adjectives *necessary* and *possible*, denoting alethic modalities, and

attitude verbs like *believe* and *hope*. We have not said anything about modal auxiliaries like *may*, *must*, *can*, and *have to*. For a more thorough and pedagogical discussion of this topic than what can be achieved here, the reader is encouraged to consult Chapter 3 of von Fintel & Heim 2011, which motivates a particular context-sensitive analysis of these elements.

[PRESENT KRATZER HERE]

13.7 Indexicals and Necessity

Let us now return to a puzzle concerning indexicals, now that we have a rudimentary treatment of intensional phenomena under our belts. Kaplan (1977) observed that the following sentence is always true, whenever uttered, and yet it does not express a necessary truth:

(27) I exist.

For most of us, anyway, it is far from necessary that we exist. Any number of circumstances could have conspired so that we never came into being. How can it be that this sentence is always true, yet not necessarily true?

Recall that in Kaplan's theory, the extension of an expression depends on a context of utterance c. Integrating this idea into our intensional semantics, the extension of an expression α will depend on a model M, and assignment g, a possible world w, and a context of utterance c.

$$[\![\alpha]\!]^{M,g,w,c}$$

The indexical constant i is defined as follows:

(28) $[\![\mathsf{i}]\!]^{M,g,w,c} = sp(c)$

The context of utterance determines not only a speaker $sp(c)$, an addressee $ad(c)$, a time of utterance $t(c)$, and a location of utterance $l(c)$, but also a designated circumstance of evaluation $w(c)$.

The designated circumstance of evaluation $w(c)$ is intuitively the world in which the utterance takes place. Truth in a context can then be defined as follows: An occurrence of ϕ in c is true iff the content expressed by ϕ in this context is true when evaluated with respect to the circumstance of the context.

The models of Kaplan's logic of indexicals determine a set of contexts, in addition to a set of individuals, a set of possible worlds, and an interpretation function. (They also contain times and positions but we will ignore those here.) So an intensional model M for a logic of indexicals would be a tuple:

$$M = \langle D, I, W, C \rangle$$

where D is a set of individuals, W is a set of possible worlds, I is a world-relative interpretation function, and C is a set of contexts. Now, there are certain constraints on these models. For example, the speaker of any context must be in the extension of the existence predicate exists at the world of the context.[3] Formally:

(29) If $c \in C$, then $sp(c) \in I_{w(c)}(\mathsf{Exists})$.

This condition on well-formed models requires that for any context c in the model, the interpretation function I must be such that the extension of the Exists predicate in the world of c contains the speaker of c. This condition guarantees that the character of 'I exist', or, formally Exists(i) will be a function from contexts to contents such that the content is true in the world of the context. In other words, for any context, the sentence will be true in the context. In this sense, 'I exist' is a LOGICAL TRUTH in Kaplan's system.

But it is not a necessary truth. Kaplan's logic of indexicals contains necessity and possibility operators defined in the standard way in modal logic. So $[\![\Box\mathsf{Exists(i)}]\!]^{M,g,w,c} = 1$ iff $[\![\mathsf{Exists(i)}]\!]^{M,g,w',c} = 1$ for all w'. If, relative to c, i denotes an individual that does not

[3] Cf. conditions 10 and 11, p. 544 of Kaplan (1977).

exist in every world, then $\Box$Exists(i) will be false. Thus Exists(i) is not a necessary truth.

Exercise 6. Explain why *I am not here now* is logically false yet not necessarily false in this framework.

Exercise 7. In this framework, pronouns and indexicals depend on different parameters of the function that assigns semantic values to expressions. Which parameter do pronouns and indexicals depend on, respectively?

Appendix

Let us take a moment to summarize what we have done. We are *almost* done with all of English, but not quite. Ha! There are extremely many topics which are fruitful to study from this perspective that we haven't touched on at all:

- comparatives: *prettier, more beautiful, more books, less pretty, fewer books, less milk*
- superlatives: *prettiest, most pretty, most books*
- exclusives: *only, sole(ly), exclusive(ly), mere(ly), just*
- exceptives: *except (for), save (that), but*
- demonstratives: *that glass over there*
- questions: *Did John kiss Mary?* and embedded questions: *John doesn't know whether he kissed Mary*
- imperatives: *Kiss Mary!*

to name a few. And there is much remaining to be said about the topics we *have* touched on. However, the reader now has a starter kit. The following sections give the fragment of English that we have developed so far.

A.1 Logic: Partial typed lambda calculus (L_3)

Expressions of the following fragment of English given below will be translated into the following version of lambda calculus in which there are three truth values. Let us call the language L_3.

Types. e and t are types, and if a and b are types, then $\langle a,b\rangle$ is a type; nothing else is a type. For all type a, $\#_a$ stands for the undefined entity of type a.

A.1.1 Syntax of L_3

The set of expressions of type a, for any type a, is defined recursively as follows. (An expression of type t is a **formula**.)

1. **Basic expressions**
   For each type a,
   (a) the **non-logical constants** of type a are they symbols of the form $c_{n,a}$ for each natural number n.
   (b) the **variables** of type a are the symbols of the form $v_{n,a}$ for each natural number n.
2. **Predication**
   For any types a and b, if α is an expression of type $\langle a,b\rangle$ and β is an expression of type a then $\alpha(\beta)$ is an expression of type b.
3. **Equality**
   If α and β are terms, then $\alpha = \beta$ is 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 $\neg\phi, [\phi\wedge\psi], [\phi\vee\psi], [\phi\rightarrow\psi]$, and $[\phi\leftrightarrow\psi]$.

6. **Quantification**
   If ϕ is a formula and u is a variable of any type, then $[\forall u.\phi]$ and $[\exists u.\phi]$ are formulas.

7. **Lambda abstraction**
   If α is an expression of type a and u is a variable of type b then $[\lambda u.\alpha]$ is an expression of type $\langle b,a\rangle$.

8. **Iota terms**
   If ϕ is a formula, and u is a variable of type a, then $[\iota u.\phi]$ is an expression of type a.

9. **Definedness conditions**
   If ϕ is a formula, then $\partial(\phi)$ is a formula.

In addition, we have the following abbreviation conventions.

1. Square brackets that are outermost in an expression may be deleted.

2. An expression of the form $[[\phi\wedge\psi]\wedge\chi]$ or $[\phi\wedge[\psi\wedge\chi]]$ can be simplified to $[\phi\wedge\psi\wedge\chi]$. Similarly for disjunctions.

3. We may write $\pi(\alpha_1,...,\alpha_n)$ instead of $\pi(\alpha_n)...(\alpha_1)$.

4. Brackets around a quantified formula can be dropped if it is rightmost (last) in a top-level expression, or rightmost in a larger constituent that ends in a bracket.

5. The dot may be dropped in a sequence of binders.

6. Square brackets that are immediately embedded inside parentheses can be dropped.

A.1.2 Semantics of L_3

For each type a, there is an associated domain D_a. D_e is the domain of individuals, D_t is the set of truth values, and for any types a and b, $D_{\langle a,b\rangle}$ is the set of functions from D_a to D_b.

Expressions are interpreted in L_3 with respect to both:

- a model $M = \langle D, I \rangle$ where D is a non-empty set of individuals, and I is a function assigning a denotation in D_a to each non-logical constant of type a
- an assignment g, which is a function assigning to each variable of type a a denotation from the set D_a

For every well-formed expression α, the semantic value of α with respect to model M and assignment function g, written $[\![\alpha]\!]^{M,g}$, is defined recursively as follows:

1. **Basic expressions**
   (a) If α is a non-logical constant, then $[\![\alpha]\!]^{M,g} = I(\alpha)$.
   (b) If α is a variable, then $[\![\alpha]\!]^{M,g} = g(\alpha)$.
2. **Predication**
   If α is an expression of type $\langle a, b \rangle$, and β is an expression of type a, then $[\![\alpha(\beta)]\!] = [\![\alpha]\!]([\![\beta]\!])$.
3. **Equality**
   If α and β are terms, then $[\![\alpha = \beta]\!]^{M,g} = 1$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g}$.
4. **Negation**
   If ϕ is a formula, then $[\![\neg\phi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 0$.
5. **Binary Connectives**
   If ϕ and ψ are formulas, then:
   (a) $[\![\phi \wedge \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$.
   (b) $[\![\phi \vee \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 1$ or $[\![\psi]\!]^{M,g} = 1$.
   (c) $[\![\phi \rightarrow \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = 0$ and $[\![\psi]\!]^{M,g} = 1$.
   (d) $[\![\phi \leftrightarrow \psi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g} = [\![\psi]\!]^{M,g}$.
6. **Quantification**

(a) If ϕ is a formula and v is a varaible of type a then $[\![\forall v.\phi]\!]^{M,g} = 1$ iff for all $k \in D_a$:

$$[\![\phi]\!]^{M,g[v \mapsto k]} = 1$$

(b) If ϕ is a formula and v is a variable of type a then $[\![\exists v.\phi]\!]^{M,g} = 1$ iff there is an individual $k \in D_a$ such that:

$$[\![\phi]\!]^{M,g[v \mapsto k]} = 1.$$

7. **Lambda Abstraction**
If α is an expression of type a and u a variable of type b then $[\![\lambda u.\alpha]\!]^{M,g}$ is that function h from D_b into D_a such that for all objects k in D_b, $h(k) = [\![\alpha]\!]^{M,g[u \mapsto k]}$.

8. **Iota terms**
If ϕ is a formula and u is a variable of type a then:

$$[\![\iota u.\phi]\!] = \begin{cases} d \; if \; \{k : [\![\phi]\!]^{M,g[u \mapsto k]} = 1\} = \{d\} \\ \#_e \text{ otherwise.} \end{cases}$$

9. **Definedness conditions**
If ϕ is an expression of type t, then:

$$[\![\partial(\phi)]\!]^{M,g} = \begin{cases} [\![\alpha]\!]^{M,g} \; if \; [\![\phi]\!]^{M,g} = 1 \\ \#_a \text{ otherwise.} \end{cases}$$

Truth in a model. For any expression ϕ, $[\![\phi]\!]^M = 1$ iff $[\![\phi]\!]^{M,g} = 1$ for every value assignment g. Similarly, $[\![\phi]\!]^M = 0$ iff $[\![\phi]\!]^{M,g} = 0$ for every value assignment g.

A.2 Syntax of English fragment

Syntax rules. The following rules derive trees at Deep Structure:

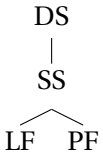
S → DP VP

S	→	S JP
JP	→	J S
VP	→	V (DP|AP|PP|CP)
AP	→	A (PP)
DP	→	(DP) D′
D′	→	D (NP)
NP	→	D (N′)
N′	→	N (PP|CP)
N′	→	A N′
PP	→	P DP
CP	→	C′
C′	→	C S

Lexicon. Lexical items are associated with syntactic categories as follows:

J:	*and, or*
Neg:	*it is not the case that*
V:	*smokes, loves, kissed, is*
A:	*lazy, proud*
N:	*drunkard, baby, kid, zebra, sister*
D:	*the, a, every, some, no, neither, 's, who, which*
	John, Obama, everybody, somebody, nobody...
P:	*of, with*
C:	*that*

Transformations. We assume the 'T-model', where a set of transformations convert Deep Structures to Surface Structures, Surface Structures to Phonological Forms, and Surface Structures to Logical Forms.



The only transformation from Deep Structure to Surface Structure that we will make explicit here is Relativization (cf. Muskens 1996):

Relativization (DS → SS). If α is *who*, *whom* or *which*:

$$[_{\mathrm{S}}\ \mathrm{X}\ [_{\mathrm{DP}}\ [_{\mathrm{D}}\ \alpha\ ]\ ]\ \mathrm{Y}\ ] \Rightarrow [_{\mathrm{CP}}\ \alpha_i\ [_{\mathrm{C}'}\ [_{\mathrm{C}'}\ \varnothing\ ]\ [_{\mathrm{S}}\ \mathrm{X}\ [_{\mathrm{DP}}\ t_i\ ]\ \mathrm{Y}\ ]\ ]\ ]$$

where i is a fresh index.

The structures that are interpreted are Logical Forms, which are derived from Surface Structures using Quantifier Raising (QR). Following May (1985), we assume that QR only allows adjunction to S nodes (whereas Heim & Kratzer (1998) allow adjunction to any expression of an appropriate semantic type), but we take the insertion of a numerical index into the tree from Heim & Kratzer (1998).

Quantifier Raising (SS → LF).

$$[_{\mathrm{S}}\ X\ [_{\mathrm{DP}}\ \alpha\ ]\ Y\ ] \Rightarrow [\ [_{\mathrm{DP}}\ \alpha\ ]\ [_{\lambda \mathrm{P}}\ i\ [_{\mathrm{S}}\ X\ [_{\mathrm{DP}}\ t_i\ ]\ Y\ ]\ ]\ ]$$

where i is a fresh index.

A.3 Translations

A.3.1 Lexical entries

We associate each lexical item with a translation to L_3. We will use the following abbreviations:

- x is $v_{0,e}$, y is $v_{1,e}$, and z is $v_{2,e}$.
- X, Y, P and Q are variables of type $\langle e,t\rangle$.
- R is a variable of type $\langle e,\langle e,t\rangle\rangle$.
- p and q are variables of type t.
- b, l, m, h and r are constants of type e.

- drunkard, baby, kid, zebra, lazy, and snores are constants of type $\langle e, t\rangle$.
- loves, kissed, with, proud, and sister are constants of type $\langle e, \langle e, t\rangle\rangle$.

Type $\langle e, t\rangle$:

1. $\langle\!\langle \textit{drunkard}\rangle\!\rangle = \lambda x.\mathsf{drunkard}(x)$
2. $\langle\!\langle \textit{baby}\rangle\!\rangle = \lambda x.\mathsf{baby}(x)$
3. $\langle\!\langle \textit{kid}\rangle\!\rangle = \lambda x.\mathsf{kid}(x)$
4. $\langle\!\langle \textit{zebra}\rangle\!\rangle = \lambda x.\mathsf{zebra}(x)$
5. $\langle\!\langle \textit{lazy}\rangle\!\rangle = \lambda x.\mathsf{lazy}(x)$

Type e:

1. $\langle\!\langle \textit{Homer}\rangle\!\rangle = \mathsf{h}$
2. $\langle\!\langle \textit{Maggie}\rangle\!\rangle = \mathsf{g}$
3. $\langle\!\langle \textit{Bart}\rangle\!\rangle = \mathsf{b}$
4. $\langle\!\langle \textit{Lisa}\rangle\!\rangle = \mathsf{l}$
5. $\langle\!\langle \textit{Marge}\rangle\!\rangle = \mathsf{m}$

Type $\langle t, \langle t, t\rangle\rangle$:

1. $\langle\!\langle \textit{and}\rangle\!\rangle = \lambda p\lambda q.[p \wedge q]$
2. $\langle\!\langle \textit{or}\rangle\!\rangle = \lambda p\lambda q.[p \vee q]$

Type $\langle t, t\rangle$:

1. $\langle\!\langle \textit{it is not the case that}\rangle\!\rangle = \lambda p.\neg p$

Type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$:

1. $\langle\!\langle \textit{is} \rangle\!\rangle = \lambda P . P$
2. $\langle\!\langle a \rangle\!\rangle = \lambda P . P$

Type $\langle \langle e, t \rangle, e \rangle$:

1. $\langle\!\langle \textit{the} \rangle\!\rangle = \lambda P . \iota x . P(x)$

Type $\langle e, \langle e, t \rangle \rangle$:

1. $\langle\!\langle \textit{loves} \rangle\!\rangle = \mathsf{loves}$
2. $\langle\!\langle \textit{kissed} \rangle\!\rangle = \mathsf{kissed}$
3. $\langle\!\langle \textit{with} \rangle\!\rangle = \mathsf{with}$
4. $\langle\!\langle \textit{proud} \rangle\!\rangle = \mathsf{proud}$
5. $\langle\!\langle \textit{sister} \rangle\!\rangle = \mathsf{sister}$

Type $\langle \langle e, t \rangle, t \rangle$:

1. $\langle\!\langle \textit{something} \rangle\!\rangle = \lambda P . \exists x . P(x)$
2. $\langle\!\langle \textit{nothing} \rangle\!\rangle = \lambda P . \neg \exists x . P(x)$
3. $\langle\!\langle \textit{everything} \rangle\!\rangle = \lambda P . \forall x . P(x)$

Type $\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$:

1. $\langle\!\langle \textit{some} \rangle\!\rangle = \lambda P \lambda Q . \exists x . [P(x) \wedge Q(x)]$
2. $\langle\!\langle \textit{no} \rangle\!\rangle = \lambda P \lambda Q . \neg \exists x [P(x) \wedge Q(x)]$
3. $\langle\!\langle \textit{every} \rangle\!\rangle = \lambda P \lambda Q . \partial [\exists x [P(x)]] \wedge \forall x [P(x) \rightarrow Q(x)]$
4. $\langle\!\langle \textit{neither} \rangle\!\rangle = \lambda P \lambda Q . [\partial [|P| = 2] \wedge \neg \exists x . [P(x) \wedge Q(x)]]$

Type $\langle e, e \rangle$:

1. $\langle\!\langle \textit{of} \rangle\!\rangle = \lambda x . x$

A.3.2 Composition rules

If the translation of an expression γ is not specified in the lexicon, then it is given by one of the following rules:

1. **Functional Application**
   Let γ be a tree whose only two subtrees are α and β. If $\langle\!\langle \alpha \rangle\!\rangle$ is of type $\langle \sigma, \tau \rangle$ and $\langle\!\langle \beta \rangle\!\rangle$ is of type σ, then:

   $$\langle\!\langle \gamma \rangle\!\rangle = \langle\!\langle \alpha \rangle\!\rangle(\langle\!\langle \beta \rangle\!\rangle)$$

2. **Predicate Modification**
   If $\langle\!\langle \alpha \rangle\!\rangle$ and $\langle\!\langle \beta \rangle\!\rangle$ are of type $\langle e, t \rangle$, and γ is a tree whose only two subtrees are α and β, then:

   $$\langle\!\langle \gamma \rangle\!\rangle = \lambda u.[\langle\!\langle \alpha \rangle\!\rangle(u) \wedge \langle\!\langle \beta \rangle\!\rangle(u)]$$

   where u is a variable of type e that does not occur free in $\langle\!\langle \alpha \rangle\!\rangle$ or $\langle\!\langle \beta \rangle\!\rangle$.

3. **Predicate Abstraction**
   If γ is an expression whose only two subtrees are α_i and β and $\langle\!\langle \beta \rangle\!\rangle$ is an expression of type t, then $\langle\!\langle \gamma \rangle\!\rangle = \lambda v_{i,e}.\langle\!\langle \beta \rangle\!\rangle$.

4. **Pronouns and Traces**
   If α is an indexed trace or a pronoun, $\langle\!\langle \alpha_i \rangle\!\rangle = v_{i,e}$

5. **Non-branching Nodes**
   If β is a tree whose only daughter is α, then $\langle\!\langle \beta \rangle\!\rangle = \langle\!\langle \alpha \rangle\!\rangle$.

We also have the following type-shifting rules:

1. **Predicate-to-modifier shift** (MOD)
   If $\langle\!\langle \alpha \rangle\!\rangle$ is of category $\langle e, t \rangle$, then:

   $$\langle\!\langle \text{MOD } \alpha \rangle\!\rangle = \lambda P \lambda x.[\langle\!\langle \alpha \rangle\!\rangle(x) \wedge P(x)]$$

   as well (as long as P and x are not free in $\langle\!\langle \alpha \rangle\!\rangle$; in that case, use different variables of the same type).

2. **Argument Raising**
   If an expression has a translation α of type $\langle \overrightarrow{a}, \langle b, \langle \overrightarrow{c}, t\rangle\rangle\rangle$, then that expression also has translations of the following form:

$$\lambda \overrightarrow{x}_{\overrightarrow{a}} \lambda v_{\langle\langle b,t\rangle,t\rangle} \lambda \overrightarrow{y}_{\overrightarrow{c}} . v(\lambda z_b[\alpha(\overrightarrow{x})(z)(\overrightarrow{y})])]$$

3. **Possessive shift**
   If $\langle\!\langle \alpha \rangle\!\rangle$ is of type $\langle e, t\rangle$, then:

$$\langle\!\langle \text{POSS } \alpha \rangle\!\rangle = \lambda y \lambda x . [\langle\!\langle \alpha \rangle\!\rangle(x) \wedge \mathsf{poss}(x, y)]$$

   as well (unless y or x is free in $\langle\!\langle \alpha \rangle\!\rangle$; in that case, use different variables of the same type).

4. **Iota shift**
   If $\langle\!\langle \alpha \rangle\!\rangle$ is of type $\langle e, t\rangle$, then

$$\langle\!\langle \text{IOTA } \alpha \rangle\!\rangle = \iota x . \langle\!\langle \alpha \rangle\!\rangle(x)$$

   as well (unless x is free in α'; then choose a different variable).

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.

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.

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.

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`.

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.

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, 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.

Cooper, Robin. 1983. *Quantification and syntactic theory*. Reidel.

Coppock, Elizabeth & David Beaver. 2015. Definiteness and determinacy. *Linguistics and Philosophy* 38(5). 377–435. doi:10.1007/s1098.

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, Robert E. Wall & Stanley Peters. 1981. *Introduction to Montague semantics*. Dordrecht: Kluwer.

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. 1999. NPI licensing, Strawson entailment, and context-dependency. *Journal of Semantics* 16. 97–148.

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.

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.

Gamut, L.T.F. 1991. *Language, logic and meaning*. Chicago University 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.

Grice, Paul. 1975. Logic and conversation. In Peter Cole & Jerry Morgan (eds.), *Syntax and semantics*, vol. 3, 41–58. New York: Academic Press.

Groenendijk, Jeroen, Theo Janssen & Martin Stokhof (eds.). 1984. *Truth, interpretation and information: selected papers from the third amsterdam colloquium*. Dordrecht, Netherlands: Foris.

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.

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. 1983c. On the projection problem for presuppositions. In M. Barlow, D. Flickinger & M. Wescoat (eds.), *Proceeding of the second annual west coast conference on formal linguistics (wccfl)*, .

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. Ms., MIT.

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.

Horn, Laurence R. 1985. Metalinguistic negation and pragmatic ambiguity. *Language* 61(1). 121–174. doi:10.2307/413423.

Hovda, Peter. 2009. What is classical mereology? *Journal of Philosophical Logic* 38(1). 55–82. doi:10.1007/s10992-008-9092-4.

Hughes, G.E. & M. J. Cresswell. 1968. *An introduction to modal logic*. London: Methuen and Co Ltd.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22. 117–184.

Jacobson, Pauline. 2000. Paycheck pronouns, Bach-Peters sentences, and variable-free semantics. *Natural Language Semantics* 8. 77–155.

Jacobson, Pauline. 2012. Direct compositionality. In *The Oxford handbook of compositionality*, 109–129. Oxford University Press.

Jespersen, Bjorn & Marie Duží. 2015. Introduction. *Synthese* 192(3). 525–534.

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.

Kaplan, David. 1989. Afterthoughts. In Joseph Almog, John Perry & Howard Wettstein (eds.), *Themes from kaplan*, 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. 1974. Presuppositions and linguistic context. *Theoretical Linguistics* 1. 181–194.

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. 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/`.

Kratzer, Angelika. 2019. Situations in natural language semantics. In Edward N. Zalta (ed.), *The stanford encyclopedia of philosophy*, Metaphysics Research Lab, Stanford University summer 2019 edition edn.

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. 1963. Semantical considerations on modal logic. *Acta Philosophica Fennica* 16. 83–89.

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.

Lewis, David. 1968. Counterpart theory and quantified modal logic. *Journal of Philosophy* 65. 113–126.

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.

Link, Godehard. 1987. Generalized quantifiers and plurals. In Peter Gärdenfors (ed.), *Generalized quantifiers: Linguistic and logical approaches*, 151–180. Dordrecht: Reidel.

Longobardi, Giuseppe. 1994. Reference and proper names: A theory of N-movement in syntax and Logical Form. *Linguistics and Philosophy* 25. 609–669.

Matthewson, Lisa. 2006. Temporal semantics in a superficially tenseless language. *Linguistics and Philosophy* 29. 673–713.

May, Robert. 1985. *Logical form: Its structure and derivation.* MIT Press.

Montague, R. 1973a. The proper treatment of quantification in ordinary English. *Approaches to natural language* 49. 221–242.

Montague, Richard. 1970. English as a formal language. In Bruno Visentini & Camillo Olivetti (eds.), *Linguaggi nella societá e nella tecnica*, vol. 87 Saggi di cultura contemporanea, 188–221. Edizioni di Comunitá.

Montague, Richard. 1973b. The proper treatment of quantification in ordinary English. In Jaakko Hintikka, Julius Moravcsik &

Patrick Suppes (eds.), *Approaches to natural language: Proceedings of the 1970 Stanford workshop on grammar and semantics*, vol. 49 Synthese Library, 221–242. Dordrecht, Netherlands: Dordrecht. doi:10.1007/978-94-010-2506-5_10.

Montague, Richard. 1979. The proper treatment of mass terms in English. In Francis Jeffry 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. Lambda grammars and hyperintensionality. Talk presented at NII, Tokyo, 19 February 2005.

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.

Parsons, Terence. 1990. *Events in the semantics of English*. Cambridge, MA: MIT Press.

Parsons, Terence. 1995. Thematic relations and arguments. *Linguistic Inquiry* 26(4). 635–662. http://www.jstor.org/stable/4178917.

Parsons, Terence. 2017. The traditional square of opposition. In Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University summer 2017 edn. https://plato.stanford.edu/archives/sum2017/entries/square/.

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.

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.

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.

Russell, Bertrand. 1905. On denoting. *Mind* 14. 479–93.

Sapir, Edward. 1944. Grading: A study in semantics. *Philosophy of Science* 11. 93–116.

Sauerland, Uli, Jan Anderssen & Kazuko Yatsushiro. 2005. The plural is semantically unmarked. In Stephan Kepser & Marga Reis (eds.), *Linguistic evidence: Empirical, theoretical and computational perspectives*, 413–434. Berlin, Germany: Mouton de Gruyter.

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.

Solan, Lawrence M. & Peter M. Tiersma. 2014. *Speaking of crime: The language of criminal justice*. University of Chicago Press.

Spector, Benjamin. 2007. Aspects of the pragmatics of plural morphology: On higher-order implicatures. In Uli Sauerland & Penka Stateva (eds.), *Presuppositions and implicature in compositional semantics*, 243–281. London, UK: Palgrave. doi:10.1057/9780230210752.

Stalnaker, Robert. 1978. Assertion. In *Syntax and semantics*, vol. 9, Academic Press.

Strawson, P. F. 1950. On referring. *Mind* 59(235). 320–344.

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.

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.

Zwarts, Frans. 1995. Nonveridical contexts. *Linguistic Analysis* 25. 286–312.