Contents

1 Implications 11

2 Sets, Relations, and Functions 25
   2.1 Sets ............................................. 25
   2.2 Ordered pairs and relations ......................... 35
      2.2.1 Ordered pairs .................................. 35
      2.2.2 Relations ..................................... 37
   2.3 Functions ......................................... 38
   2.4 Negative polarity items ............................... 42

3 First order logic 51
   3.1 Atomic formulas .................................... 52
      3.1.1 Individual constants .............................. 52
      3.1.2 Predicate symbols and atomic formulas .......... 53
      3.1.3 Function symbols ................................ 57
      3.1.4 Boolean connectives .............................. 60
      3.1.5 Conditionals .................................... 65
   3.2 Equivalence and tautology ......................... 67
   3.3 Summary: $L_0$ ...................................... 69
      3.3.1 Syntax of $L_0$ .................................. 70
      3.3.2 Semantics of $L_0$ ................................ 71
   3.4 Quantification ..................................... 72
      3.4.1 Syntax of $L_1$ .................................. 81
      3.4.2 Semantics of $L_1$ .............................. 85
# Contents

## 4 Typed lambda calculus

- 4.1 Lambda abstraction .................................................. 93
- 4.2 Summary .............................................................. 103
  - 4.2.1 Syntax of $\lambda$ .................................................. 103
  - 4.2.2 Semantics .......................................................... 110

## 5 Translating to lambda calculus

- 5.1 Fun with Functional Application ................................. 119
  - 5.1.1 Homer loves Maggie ............................................. 119
  - 5.1.2 Homer is lazy ...................................................... 122
  - 5.1.3 Homer is with Maggie .......................................... 124
  - 5.1.4 Homer is proud of Maggie .................................... 126
  - 5.1.5 Homer is a drunkard ............................................ 127
  - 5.1.6 Toy fragment ..................................................... 131
- 5.2 Predicate Modification ............................................. 133
  - 5.2.1 Homer is a lazy drunkard .................................... 133
- 5.3 Quantifiers ........................................................... 144
  - 5.3.1 Quantifiers: not type $e$ ..................................... 144
  - 5.3.2 Solution: Quantifiers ......................................... 148
- 5.4 The definite article .................................................. 155

## 6 Variables in Natural Language

- 6.1 Relative clauses ..................................................... 167
- 6.2 Quantifiers in object position ................................... 179
  - 6.2.1 Quantifier Raising ............................................. 179
  - 6.2.2 A type-shifting approach .................................... 186
  - 6.2.3 Putative arguments for the movement .................... 191
- 6.3 Possessives .......................................................... 199
- 6.4 Pronouns ............................................................. 206

## 7 Dynamic semantics

- 7.1 Pronouns with indefinite antecedents .......................... 215
- 7.2 File change semantics .............................................. 222
- 7.3 Discourse Representation Theory ................................. 226
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3.1 Introducing Intensional Logic</td>
<td>324</td>
</tr>
<tr>
<td>11.3.2 Formal fragment</td>
<td>329</td>
</tr>
<tr>
<td>11.4 Fregean sense and hyperintensionality</td>
<td>333</td>
</tr>
<tr>
<td>11.5 Explicit quantification over worlds</td>
<td>335</td>
</tr>
<tr>
<td>11.5.1 Modal auxiliaries</td>
<td>336</td>
</tr>
<tr>
<td>11.6 Logic of Indexicals</td>
<td>336</td>
</tr>
<tr>
<td>12 Tense</td>
<td>347</td>
</tr>
</tbody>
</table>
1 Implications

Semantics and pragmatics are two subfields of linguistics that deal with meaning. What is meaning? What is it to understand? For instance, does Google understand language? Many might argue that it does, in some sense. One could point to the fact that one can type in, “350 USD in SEK” and get back for example “2232.23 Swedish Krona” as the first result. This is a step toward understanding. But it can be argued that in general, Google does not understand language because it does not do inferences. For example, there is a webpage that says:

(1) Beth speaks English, French and Spanish.

If we were to ask Google, “Does Beth speak German?”, Google would not know the answer. It wouldn’t even know the answer if we asked, “Does Beth speak a European language?”

A hallmark of a system or agent that understands language or grasps meaning is that it can do these kinds of inferences. Another way to put this is that a good theory of meaning should be able to explain when one sentence implies another sentence. The implication relations that hold between sentences are perhaps the main kind of facts that semantics and pragmatics try to explain. (Semantics and pragmatics can also be used to explain why certain sentences sound odd to native speakers, i.e., acceptability judgments, and facts about frequency in natural language corpora.)

What exactly does ‘imply’ mean? In other words, what are im-
Implications

Implication can be defined either as a relation between two ideas (or propositions) or as a relation between two sentences. (Other options are also possible.) We will define it as a relation between two sentences here. The definition should be weak enough that it will cover all of the different types of implications that we will discuss. It should also take into account the fact that whether or not one sentence implies another can depend on the context in which it is spoken. Let us therefore use the following definition:

\[
(2) \quad \text{A sentence } \phi \text{ IMPLIES another sentence } \psi \text{ in context } c \text{ if and only if:}
\]

By expressing \( \phi \) in context \( c \), a speaker signals that \( \psi \) is true.

We use \( \phi \) ‘phi’ and \( \psi \) ‘psi’ as placeholders for sentences, following the logical tradition of using Greek letters for variables that range over expressions in the language that is being theorized about.

To apply the definition, we plug sentences in for these variables. For example: In ordinary contexts, \( \text{Infer } \) implies \( \text{Beth does not speak German} \), because by expressing \( \text{Beth speaks English, French and Spanish} \), the author of the website signals that \( \text{Beth does not speak German} \) is true.

---

1 Terminological note: An implication (or implication relation) is a relation that holds between a premise and an implied conclusion. Strictly speaking, the noun *inference* describes an 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*; an intelligent person *infers* a conclusion from a premise, but a premise *implies* a conclusion. The subject of *infer* is the person doing the inference, but the subject of *imply* is the premise.

2 In general, the term *object language* is used to refer to the language that we are talking about. So \( \phi \) and \( \psi \) in this case range over sentences of the object language. The language in which we theorize about the object language is called the *meta-language*. Since the Greek letters \( \phi \) and \( \psi \) are part of the meta-language, we can call them *meta-language variables* (even though they range over expressions of the object language). When we get to formal logic, we will encounter variables that are part of the object language, i.e., *object language variables*. 
There are several different types of implications. In conjunction with the fact that French is a European language, logically implies that Beth speaks a European language. This is an example of an entailment. We define entailment as follows.

(3) A sentence $\phi$ entails a sentence $\psi$ if and only if:
Whenever $\phi$ is true, $\psi$ is true too.

(Entailment is not usually thought of as a context-sensitive relation, although there is a notion of contextual entailment, which holds between two sentences if the first, combined with the information that is already given in the context, entails the second. Moreover, the semantic content of a sentence may depend on features of the context in which it is uttered, such as who is speaking, and when. A more careful definition would take this into account.) In the following pairs, the (a) sentence entails the (b) sentence:

(4) a. All cars are blue.
   b. All sportscars are blue.

(5) a. Mary invited Fred and Jack.
   b. Mary invited Fred.

   b. Jane graduated from MIT.

(7) a. There are three pens on the table.
   b. More than two pens are on the table.

Each of these examples satisfies the definition of entailment, at least against the background of very basic assumptions. For example, given that all sportscars are cars, All cars are blue entails All sportscars are blue, because in every imaginable world or situation where the former is true, the latter is true too. Whenever There are three pens on the table is true, More than two pens are on the table is true too.
Note that there are several other ways that entailment can be defined (Chierchia & McConnell-Ginet, 2000):

(8) **Alternative definitions of entailment from \( \phi \) to \( \psi \)**

a. The information that \( \psi \) conveys is contained in the information that \( \phi \) conveys.

b. A situation describable by \( \phi \) must also be a situation describable by \( \psi \).

c. \( \phi \) and it is not that \( \psi \) is contradictory (can't be true in any situation).

For example, the information that *More than two pens are on the table* conveys is contained in the information that *There are three pens on the table* conveys; a situation describable by *There are three pens on the table* must also be a situation describable by *More than two pens are on the table*, and *There are three pens on the table and it is not true that more than two pens are on the table* cannot be true in any situation.

Example (1) also implies that Beth does not speak German. But would you say that the sentence is false if she did? 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 Beth speaks. If she did speak German, the speaker would be “lying by omission”. This 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 the norms of conversation (Grice, 1975).

If I said, *Beth no longer speaks French*, I would signal a **PRESUPPOSITION** that she spoke French at some time in the past. To presuppose something is to take it for granted, to treat it as uncontroversial and known to everyone participating in the conversation. *No longer* is among the set of words and phrases that can be used to signal presuppositions. Presupposition can be seen as a type of entailment: In every situation where *Beth no longer speaks French* is true, *Beth spoke French at some time in the past* is also true. But
Implications

presuppositions differ from ordinary entailments, as you can see from what happens when they are negated or put in the form of a question. For example, suppose someone were to ask, Does Beth no longer speak French? This would imply that Beth spoke French at some time in the past. The same pattern does not arise with ordinary entailment. Merely asking Does Beth speak French? does not imply that Beth speaks a European language.

Semantics is sometimes said to be the study of what linguistic expressions mean, while pragmatics is the study of what speakers mean by them. 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 in 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, we will focus primarily on ordinary entailments, but presuppositions will be addressed as well.

If we know the conditions under which two given sentences are true, then we can determine, given the definition of entailment above, whether one entails the other. We don’t need to know whether either sentence is in fact true; we only need to be able to discriminate between situations in which the sentence is true and situations in which the sentence is false. The truth conditions are the conditions under which the sentence is true.

It would not be prudent to claim that meaning of a sentence in natural language consists entirely in its truth conditions, although this view is sometimes expressed. Heim & Kratzer (1998) begin their book with the following bold sentence: “To know the meaning of a sentence is to know its truth conditions.” Many find this view objectionable; meaning is not just truth conditions. The non-entailment implications that a sentence has can also be considered part of its meaning. There are other aspects of meaning that truth conditions leave out as well. For example, arguably, 2 +
Implications

2 = 4 is true under just the same circumstances as Dirichlet’s Theorem is true (namely always, since they are mathematical truths and therefore do not vary in their truth value from situation to situation). Yet it is possible to know one without knowing the other. This phenomenon goes by the name of hyperintensionality, and is related to Frege’s (1892) famous distinction between ‘sense’ and ‘reference’. Such phenomena show that there is more to meaning than truth conditions. But truth conditions are a way of capturing one important aspect of meaning, namely entailment. If the truth conditions for $\psi$ are satisfied whenever the truth conditions for $\phi$ are satisfied, then we know that $\phi$ entails $\psi$.

The strategy we follow has its roots in the study of logic, which has an extremely long recorded history. Notable developments include the work of Aristotle, whose syllogisms were studied for centuries. A syllogism can be defined as a form of argument consisting of one or more premises and a conclusion, where the conclusion necessarily follows from the premises. The most famous example is:

\begin{align*}
\text{(9) } & \text{All men are mortal. (Premise 1)} \\
& \text{Socrates is a man. (Premise 2)} \\
& \text{Therefore, Socrates is mortal.}
\end{align*}

An argument in which the conclusion follows from the premises is called a valid argument. In other words, a valid argument is one such that the premises, taken together, entail the conclusion. Note that the premises of an argument need not be true in order for the argument to be valid; an argument is valid as long as the conclusion is entailed by the premises. A sound argument is one whose premises are true. An argument may be sound, but not valid, valid but not sound, both (ideally), or neither (unfortunately).

Socrates’ interest in the form of valid arguments made him the first formal logician. Aristotelian logic and developments thereof enjoyed a rich and intensive period of study during the Middle
Ages, but it died out at the hands of the Renaissance humanists in the 14th century. Interest in logic was revived in the 19th century by the likes of Gottlieb Frege and Bertrand Russell, who wanted to set mathematics on a firm logical footing. To do so, Frege and Russell developed artificial languages, which, unlike natural languages, are completely unambiguous. In the preface to his *Begriffsschrift*, Frege wrote:

> If the task of philosophy is to break the domination of words over the human mind [...] then my concept notation, being developed for these purposes, can be a useful instrument for philosophers [...] I believe the cause of logic has been advanced already by the invention of this concept notation.

As Frege’s primary interest was mathematical, his artificial notation was not specifically intended as a way to explicate the meanings of sentences in natural language. Indeed, up until the 1960’s there was a broad consensus among philosophers of language from otherwise diverse camps that there was an intractable divide between formal logic and natural language. Noam Chomsky, despite his application of formal language theory to natural language syntax, was also deeply skeptical of the relevance of formal logic to linguistic theory.

But in the 1970’s Richard Montague (among others) dared to bridge this divide, beginning one of his most famous articles with the bold claim: “I reject the contention that an important theoretical difference exists between formal and natural language.” Montague defined a ‘fragment’ of English just as formally and precisely as a formal logic, laying the groundwork for a new approach. The potential of this approach might never have been explored nearly as fully as it has been if it hadn’t been for Barbara Partee, a linguist who took an interest in his approach, advocated for it, and extended its frontier. Thanks to her efforts, and those of her students, formal semantics has become a core area of generative linguistics.
This book will present a modern version of Montague's approach. To do this, we will use several ingredients. **First**, we need to make certain assumptions about the nature of what language describes – what makes up reality and dreams and lies. For this, semanticists typically borrow ideas and notation from set theory, which will be discussed in the next chapter. Set theory will thus form part of our meta-language. Using this precise meta-language, we can start describing the circumstances with respect to which a given sentence is true. For example, we could imagine a circumstance where Beth speaks only Turkish, and under that circumstance, sentence (1) is false. The class of circumstances under which a sentence is true can be identified with the truth conditions of a sentence: the circumstances where a sentence is true are those under which the truth conditions are satisfied. These circumstances can be called **circumstances of evaluation**, and are often conceived of as possible worlds or situations. A possible world is just a way that the world might be; a situation typically carves out a particular part of a possible world. A model for first-order logic can be used to represent a possible world, and our models will be characterized using ideas from set theory.

**The second ingredient** we will use is an unambiguous formal language that has a clearly defined relationship to these models. To define such a language, we borrow tools from formal logic. This formal language is *not* the meta-language (although it is not uncommon to hear semanticists and philosophers speak this way), nor should it be called 'Logical Form'. We reserve the term 'Logical Form' for an abstract level of syntactic representation for natural language; we will get to that later. As the formal language serves as an intermediary between the natural language and the truth conditions, representing the latter precisely, we call it a (**logical**)

**Secondly**, we define rules connecting expressions of natural language to this formal language. Once the natural language is connected to the formal language, and the formal language is con-
nected to the models, we can determine when one natural language sentence entails another, and what the truth conditions of natural language sentences are.

Crucially, the system that we build will be compositional. This means that the meaning of a larger expression can be determined from the meanings of the parts. This means that our rules for connecting expressions of natural language to formal language must include not only a way of assigning meanings to simple words and other atomic lexical units (a lexicon), but also a way of combining these meanings together to assign meanings to complex units. The way we do this is using composition rules. One of the deeper questions that research in semantics engages with is how many composition rules are necessary. According to what Heim & Kratzer (1998) call Frege’s Conjecture, we need only one rule: Functional Application. Advocates of Construction Grammar emphasize phenomena that seem to point to a need for many different composition rules, corresponding to different constructions. In this book, we will present the moderately slim theory of composition given by Heim & Kratzer (1998), consisting of only five very general-purpose composition rules:

- Functional Application
- Predicate Modification
- Predicate Abstraction
- Lexical Terminals
- Pronouns and Traces

The first three are for putting together complex expressions; the last two are for interpreting simple, atomic units.

Note that insofar as we are first defining a formal language and translating expressions of natural language into it, the system presented here is similar to the one in Richard Montague’s famous paper, ‘The Proper Treatment of Quantification in Ordinary English,’
Implications

affectionately known as ‘PTQ’. This method is known as INDIRECT INTERPRETATION. Another way to go about things would be to skip the formal language and give the interpretations of English expressions directly using our meta-language, as Montague did in his paper ‘English as a Formal Language’. That style is known as DIRECT INTERPRETATION. Using indirect interpretation is one respect in which this textbook differs from that of Heim & Kratzer (1998). Indirect interpretation offers a number of practical technical advantages, and meshes well with the Lambda Calculator, pedagogical software that can be used in conjunction with this book.

The system presented here is quite simple. There are a number of interesting ways in which it falls short at explaining natural language phenomena. Don’t be discouraged by this. One reason to learn it is that it can be used as a lingua franca with other linguists since it is quite standard. Another reason not to despair too quickly is that after the theory is modified and extended to capture more and more data, it may look very little like how it did originally. Consider this the basic starter kit for a theory of semantics. If you understand the foundations well (and you have a good supplier of bells and whistles), you will be able to modify it to suit your purposes.

Exercise 1. Consider the following exchange that took place when American movie producer Samuel Bronston was being questioned under oath during a bankruptcy hearing:

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 Zurich.

It was later revealed that Bronston had had a Swiss bank account
Implications

for about 5 years. Clearly, Bronston’s answer implied, in some sense of *imply*, that he had not personally ever had a Swiss bank account. Is this implication an entailment? Explain your answer.

**Exercise 2.** One of the hallmarks of presuppositions is that they persist under negation and in questions. For example, whether I say *Mary has stopped smoking*, *Mary hasn’t stopped smoking*, or *Has Mary stopped smoking?* I imply that Mary has smoked in the past. This is not true of entailments. For example, an entailment of *Mary owns a golden retriever* is *Mary owns a dog*, and this does not persist under negation; consider *Mary does not own a golden retriever* and *Does Mary own a golden retriever?* Use this *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 3.** Two sentences have the same truth conditions if they are true under exactly the same circumstances. Do the following pairs of sentences have the same truth conditions, i.e., are they logically equivalent? Explain your reasoning.

(a) I know a bachelor.
   I know an unmarried male.

(b) John likes Mary but he fears her.
   John likes Mary and he fears her.
Exercise 4. Which of the following are valid? Which are sound?

(a) Every Spaniard is female; George Bush is a Spaniard; therefore, George Bush is female.

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

(c) Every Swede is a socialist; Bernie Sanders is a socialist; therefore Bernie Sanders is a Swede.

(d) If $2 + 2 = 5$ then the moon is made of green cheese; $2 + 2 = 5$; therefore, the moon is made of green cheese.

Exercise 5. 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 an example of such an argument. If your answer is no, explain why.

Exercise 6. Explain the difference between meta-language and object language.
Exercise 7. What are the three ingredients of the formal theory to be presented in this book? Use as few words as possible in your answer.
Recall that one of the ingredients of our formal theory is a way of modelling the realities, dreams, and lies that language depicts. To achieve this, we need to introduce certain tools for describing the circumstances under which a given sentence is true or false. In this chapter, we will introduce the most fundamental of these tools: sets, ordered pairs and relations, and functions.

### 2.1 Sets

A set is an abstract collection of distinct objects which are called the members or elements of that set. 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. Notice that the members of the list are separated by commas and enclosed by curly braces. To denote the fact that 2 is a member of this set, we can write:

\[ 2 \in \{2, 7, 10\} \]

To denote the fact that 3 is not a member of this set, we can write:

\[ 3 \notin \{2, 7, 10\} \]

This statement can be read, ‘3 is not an element of the set containing 2, 7 and 10.’
Note that 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\}

Note also that listing an element multiple times does not change the membership of the set. Thus:

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

is exactly the same set as this set:

\{3\}

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

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

The ellipsis notation (...) 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 \textsc{singleton}. A set can even be empty. The set with no elements at all is called the \textsc{empty set}, written either like this:

\{
\}

or like this:

\∅

The \textsc{cardinality} of a set is the number of elements it contains. The cardinality of the empty set, for example, is 0. Cardinality is
denoted with vertical bars surrounding the set: If $A$ is a set, then $|A|$ denotes 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:

$$\{\emptyset, 2\}$$

This set has two elements, not one.

**Exercise 1.** How many elements do the following sets contain?

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

(b) $\emptyset$

(c) $\{\emptyset\}$

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

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

In the kind of set theory that linguists typically use, elements may be either concrete (like the beige 1992 Toyota Corolla I sold in 2008, you, or your computer) or abstract (like the number 2, the English phoneme /p/, or the set of all Swedish soccer players). 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, and similarly the set of all former first-grade teachers is perfectly determined, 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 to describe the set of things meeting a certain condition. To do that, we place a variable – a name 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. For example, to describe the set of numbers below zero, we can introduce the variable \( n \) (for ‘number’) \( n < 0 \) to describe the condition that any number \( n \) must meet in order to be counted as part of the set. The result looks like this:

\[
\{ n \mid n < 0 \}
\]

This expression can be read, ‘the set of numbers \( n \) such that \( n \) is less than 0’. So the vertical bar is pronounced ‘such that’ in this context.

The set \( \{2, 3\} \) is not an element, but rather a subset of the set \( \{2, 3, 4\} \). In general, a set \( A \) is a subset of a set \( B \) if and only if every member of \( A \) is a member of \( B \). Put more formally:

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

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, with the dashed circle for the snorers and the plain circle for the musicians.

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:

\[
\{a, \{b\}\} \notin \{a, b, c\}
\]

(The slash across the \(\subseteq\) symbol negates it, so \(\notin\) can be read ‘is not a subset of’.) The reason \(\{a, \{b\}\}\) is not a subset of \(\{a, b, c\}\) is that the former has a member that is not a member of the latter, namely \(\{b\}\). It is tempting to think that \(\{a, \{b\}\}\) contains \(b\) 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. 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:

\[a \in \{a, \{b\}, c\}\]

\[\{b\} \in \{a, \{b\}, c\}\]
Note that the empty set is a subset (not an element!) of every set. So, in particular:

\[ \emptyset \subseteq \{ a, b, c \} \]

Since the empty set doesn’t have any members, it never contains anything that is not part 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 the set in order to decide.)

Note also that by this definition, every set is actually a subset of itself, even though normally we think of two sets of different sizes when we think of the subset relation. So:

\[ \{ a, b, c \} \subseteq \{ a, b, c \} \]

To avoid confusion, it helps to distinguish between subsets and proper subsets. \( A \) is a proper subset of \( B \), written \( A \subset B \), if and only if \( A \) is a subset of \( B \) and \( A \) is not equal to \( B \).

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

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

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

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

And as you might expect, \( A \) is a proper superset of \( B \), written \( A \supset B \), if and only if \( A \) is a superset of \( B \) and \( A \) is not equal to \( B \).

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

The intersection of \( A \) and \( B \), written \( A \cap B \), is the set of all entities \( x \) such that \( x \) is a member of \( A \) and \( x \) is a member of \( B \).
\[ A \cap B = \{ x \mid x \in A \text{ and } x \in B \} \]

For example:

\[
\{a, b, c\} \cap \{b, c, d\} = \{b, c\} \\
\{b\} \cap \{b, c, d\} = \{b\} \\
\{a\} \cap \{b, c, d\} = \emptyset \\
\{a, b\} \cap \{a, b\} = \{a, b\}
\]

Intersection is very useful in natural language semantics. It can be used as the basis for a semantics of \textit{and}. For example, if someone tells you that John is a lawyer \textit{and} a doctor, then you know that John is in the \textit{intersection} between the set of lawyers and the set of doctors. If the dashed circle in the following diagram represents doctors, and the plain circle represents lawyers, then John is located somewhere in the area where the two circles overlap, if he is both a doctor and a lawyer.

![Diagram showing intersection of two sets]

The English determiner \textit{some} could be thought of in terms of intersection as well, as a relation between two sets \(X\) and \(Y\) which holds if there is some member of \(X\) which is also a member of \(Y\), i.e., if the intersection between \(X\) and \(Y\) is non-empty. For instance, \textit{some musician snores} should be true if there is some individual which is both a musician and a snorer.

\textit{No} can be thought of as a relation between two sets \(X\) and \(Y\) which holds if the two sets have no members in common, in other words, if the \textit{intersection is empty}. So \textit{no musician snores} holds if there is no individual who is both a musician and a snorer. In that case, the two sets are \textit{disjoint}, like so:
Another useful operation on sets is union. The union of $A$ and $B$, written $A \cup B$, is the set of all entities $x$ such that $x$ is a member of $A$ or $x$ is a member of $B$.

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

For example:

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

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

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

As the reader 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.)

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 entities $x$ such that $x$ is an element of $A$ and $x$ is not an element of $B$.

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

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.
Exercises on sets

The following exercises are taken from Partee, ter Meulen and Wall, *Mathematical Methods in Linguistics*.

**Exercise 2.** Given the following sets:

\[ A = \{ a, b, c, 2, 3, 4 \} \quad E = \{ a, b, \{ c \} \} \]
\[ B = \{ a, b \} \quad F = \emptyset \]
\[ C = \{ c, 2 \} \quad G = \{ \{ a, b \}, \{ c, 2 \} \} \]
\[ D = \{ b, c \} \]

classify each of the following statements as true or false.

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

**Exercise 3.** Consider the following sets:

\[ S_1 = \{ \emptyset, \{ A \}, A \} \quad S_6 = \emptyset \]
\[ S_2 = A \quad S_7 = \{ \emptyset \} \]
\[ S_3 = \{ A \} \quad S_8 = \{ \emptyset \} \]
\[ S_4 = \{ \{ A \} \} \quad S_9 = \{ \emptyset, \{ \emptyset \} \} \]
\[ S_5 = \{ \{ A \}, A \} \]

(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 4.** Given the sets $A, ..., G$ from above, repeated here:

\[
A = \{a, b, c, 2, 3, 4\} \quad E = \{a, b, \{c\}\} \\
B = \{a, b\} \quad F = \emptyset \\
C = \{c, 2\} \quad G = \{\{a, b\}, \{c, 2\}\} \\
D = \{b, c\}
\]

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

(a) $A \cup B$

(b) $A \cap B$
2.2 Ordered pairs and relations

The meanings 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. Technically, a relation is a set of *ordered pairs*.

2.2.1 Ordered pairs

As stated above, sets are not ordered:

\[ \{ a, b \} = \{ b, a \} \]

But the elements of an ordered pair written \( \langle a, b \rangle \) are ordered. Here, \( a \) is the first member and \( b \) is the second member.

\[ \langle a, b \rangle \neq \langle b, a \rangle \]

An ordered pair always consists of two members, a first member and a second member. But like sets, the members of an ordered pair can be almost 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 \( \{a, b, c\} \) and whose second member is the set \( \{d, e, f\} \), written:

\[ \langle \{a, b, c\}, \{d, e, f\} \rangle \]

Alternatively, one or both of the members could be an ordered pair, 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 \( \langle 10, 12 \rangle \).

Exercise 6. 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) \( \{\{3, 3\}\} = \langle 3, 3 \rangle \)

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

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

(h) \( \{3, \{3, 4\}\} = \{3, \{4, 3\}\} \)
2.2.2 Relations

As mentioned above, transitive verbs like love, admire, and respect are sometimes thought of as relations between two individuals. Certain nouns like mother can also be thought of as expressing relations. The ‘love’ relation corresponds to the set of ordered pairs of individuals such that the first member loves the second member. Suppose John loves Mary. Then the pair \( \langle \text{John}, \text{Mary} \rangle \) is a member of this relation.

A relation has a domain and a range. The domain is the set of objects from which the first members are drawn, and the range is the set of objects from which the second members are drawn. If \( R \) is a relation between two sets \( A \) and \( B \), then we say that \( R \) is a relation from \( A \) to \( B \). For example, the mother relation might be thought of as a relation from female animals to animals.

Sometimes it is useful to talk about the relation that results from pairing every element of one set with every element of another set. The Cartesian product of two sets \( A \) and \( B \), written \( A \times B \), is the set of ordered pairs that can be constructed by pairing a member of \( A \) with a member of \( B \).

\[
A \times B = \{ \langle x, y \rangle \mid x \in A \text{ and } y \in B \}
\]

For example, the Cartesian product of \( \{a, b, c\} \) and \( \{1, 0\} \) is:

\[
\{ \langle a, 1 \rangle, \langle a, 0 \rangle, \langle b, 1 \rangle, \langle b, 0 \rangle, \langle c, 1 \rangle, \langle c, 0 \rangle \}
\]

If \( R \) is a relation from \( A \) to \( B \), then \( R \) is a subset of the Cartesian product of \( A \) and \( B \). In symbols:

\[
R \subseteq A \times B
\]
Exercise 7. All of the following are relations from the set containing the Nordic countries (Iceland, Sweden, Norway, Finland, and Denmark) to \{0, 1\}:

\[ R_1: \{(\text{Sweden}, 0), (\text{Norway}, 0), (\text{Iceland}, 1), (\text{Denmark}, 1), (\text{Finland}, 1)\} \]

\[ R_2: \{(\text{Sweden}, 1), (\text{Norway}, 1), (\text{Iceland}, 0), (\text{Denmark}, 0), (\text{Finland}, 0)\} \]

\[ R_3: \{(\text{Sweden}, 1), (\text{Norway}, 1)\} \]

\[ R_4: \{(\text{Sweden}, 1), (\text{Norway}, 1), (\text{Iceland}, 0), (\text{Denmark}, 0), (\text{Finland}, 0), (\text{Sweden}, 0), (\text{Norway}, 0)\} \]

\[ R_5: \{\text{Sweden, Norway, Iceland, Denmark, Finland}\} \times \{0, 1\} \]

Questions:

(a) All of these relations have the same domain. What is it?

(b) All of these relations have the same range. What is it?

(c) True or false:

All of these relations are subsets of the Cartesian product of \{Iceland, Sweden, Norway, Finland, Denmark\} and \{0, 1\}.

2.3 Functions

Now we are ready to introduce functions. Intuitively, a function is 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 chocolate bar). (Let us set aside the fact that vending machines typically also require an input of money.) This kind of idea can be treated mathematically as a relation from the set of inputs to
the set of outputs. Importantly, for something to be a function, it must be predictable exactly which output you are going to get for a given input. So every input should be paired with one and only one output. 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](image)

Formally, a function is a special kind of relation. (Very special! Functions lie at the heart of modern formal semantics.) A relation $R$ from $A$ to $B$ is a **function** if and only if it meets both of the following conditions:

- Each element in the domain is paired with just one element in the range.
- The domain of $R$ is equal to $A$.

For example, $\{\langle 0,0 \rangle, \langle 0,1 \rangle \}$ is not a function because given the input 0, there are two outputs: 0 and 1.
Exercise 8. Which of the relations $R_1 - R_5$ are functions? It may help to draw them.

We write $F(a)$ to denote ‘the result of applying function $F$ to argument $a$’ or $F$ of $a$’ or ‘$F$ applied to $a$’. If $F$ is a function that contains the ordered pair $(a, b)$, 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 9. Given $R_1$ as defined above:

(a) What value does $R_1$ take, given Norway as an argument?
(b) In other words, what is $R_1$(Norway)?

(c) What is $R_1$(Iceland)?

(d) What is $R_2$(Norway)?

**Characteristic function.** As mentioned above, the semantics of common nouns like *tiger* and *picnic* and *student* are sometimes thought of in terms of sets – the set of tigers, the set of picnics, the set of students. But another way of treating their meaning (which will prove useful to us) is as functions from individuals to truth values. For example, the set of children in the Simpsons family is \{Bart, Maggie, Lisa\}. One possible analysis of the word *child* relative to the Simpsons domain is this set. But we could also treat it as a function that takes an individual and returns a truth value (0 or 1; 0 for “false” and 1 for “true”):

\[
\{\langle Homer, 0 \rangle, \langle Marge, 0 \rangle, \langle Bart, 1 \rangle, \langle Lisa, 1 \rangle, \langle Maggie, 1 \rangle\}
\]

Here is an alternative way of writing the same thing:

\[
\begin{array}{l}
\text{Homer} \to 0 \\
\text{Marge} \to 0 \\
\text{Bart} \to 1 \\
\text{Lisa} \to 1 \\
\text{Maggie} \to 1 \\
\end{array}
\]

This function, applied to Lisa, would yield 1 (“true”). Applied to Homer, it would yield 0 (“false”), because Homer is not a child (despite the fact that Lisa is more mature than he is). A function that yields 1 (“true”) for every input that is in set $S$ is called the **characteristic function** of $S$. The function we have just been looking at is the characteristic function of the set \{Bart, Lisa, Maggie\}. 
Exercise 10.

(a) What set is \( R_1 \) is the characteristic function of? (You can define it by listing the elements.)

(b) What set is \( R_2 \) is the characteristic function of?

(c) Give the characteristic function of the set of Scandinavian countries, with the set of Nordic countries as its domain.

2.4 Negative polarity items

With some basic tools from set theory, we can start to get a grip on one of the classic puzzles of semantics. Here is the puzzle. There are certain words of English, including *any*, *ever*, *yet*, and *anymore*, which can be used in negative sentences but not positive sentences:

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

These are called **NEGATIVE POLARITY ITEMS** (NPIs). It’s not just negative environments where NPIs can be found. Here is a sampling of the data [Ladusaw 1980].

(2)  
\[
\begin{align*}
\text{No one} \\
\text{At most three people} \\
\text{Few students} \\
\text{*Someone} \\
\text{*At least three people} \\
\text{*Many students}
\end{align*}
\]

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. John finished his homework \{ without
*with \} any help.

b. John voted \{ against
*for \} ever approving any of the proposals.

(5) John will replace the money \{ before
if
*after
*when \} anyone ever misses it.

(6) It's \{ hard
difficult
*easy
*possible \} to find anyone who has ever read anything much about phrenology.

(7) John \{ doubted
denied
*believed
*hoped \} that anyone would ever discover that the money was missing.

(8) It \{ *is likely
*is certain
is surprising
*is unsurprising \} 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.

The issue is made a bit more complex by the fact that words differ as to *where* they license negative polarity items. *No* licenses NPIs throughout the sentence:

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

But the determiner *every* licenses NPIs only in the noun phrase immediately following it:

(10)  
   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.

*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*:

(11)  
   a. Mary is a cellist.
   b. ⇒ Mary is a musician.

We can describe this situation in terms of sets and subsets. If we think of these terms as being arranged visually in a taxonomic hierarchy with more specific concepts at the bottom, we can say that the inference in (7) proceeds from lower (more specific) to higher (more general), hence “upwards”.
An entailment by a sentence of the form [...] to a sentence of the form [...] where $A$ is more specific than $B$ can thus be labelled an **upward entailment**. Here is another upward entailment:

(12) a. Some cellists snore.
    b. $\Rightarrow$ Some musicians snore.

But negation and the determiner *no* reverse the entailment pattern:

(13) a. Mary isn’t a musician.
    b. $\Rightarrow$ Mary isn’t a cellist.

(14) a. No musicians snore.
    b. $\Rightarrow$ No cellists snore.

These entailments are called, as the reader may have guessed, **downward entailments**, because they go from more general (higher) to more specific (lower).

There is a correlation between NPI-licensing and downward entailment: NPIs occur where downward entailments occur. Compare the following examples to the NPI data for *no*, *some* and *every* above.

(15) a. No musician snores.
    $\Rightarrow$ No cellist snores. (downward)
b. No musician snores.
   \[\Rightarrow \text{No musician snores loudly.}\] (downward)

(16) a. Every musician snores.
   \[\Rightarrow \text{Every cellist snores.}\] (downward)

b. Every musician snores loudly.
   \[\Rightarrow \text{Every musician snores.}\] (upward)

(17) a. Some cellists snore.
   \[\Rightarrow \text{Some musicians snore.}\] (upward)

b. Some musician snores loudly.
   \[\Rightarrow \text{Some musician snores.}\] (upward)

Ladusaw’s generalization was as follows: An expression licenses negative polarity items in its scope if it licenses downward entailments in its scope. The “scope” of an expression is the constituent it combines with syntactically. We can assume that a determiner like no, every, or some combines syntactically with the noun next to it, and that the resulting noun phrase combines syntactically with the verb phrase, and the following syntactic structure in particular:\[2\]

\[
\begin{array}{c}
\text{S} \\
\text{NP} \quad \text{VP} \\
\text{D} \quad \text{N'} \\
\mid \quad \text{snores} \\
\text{no} \quad \text{musician}
\end{array}
\]

So no licenses NPIs in the N', and the expression no musician licenses NPIs and downward entailments in the VP. Every licenses

---

\[2\] S stands for “Sentence”, NP stands for “Noun Phrase”, VP stands for “Verb Phrase”, D stands for “Determiner”, and N' should be read “N-bar”; it stands for an intermediate phrase level in between nouns and noun phrases. Do an internet search for “X-bar syntax” to find out more about it. The triangles in the trees indicate that there is additional structure that is not shown in full detail.
NPIs in the $N'$, but the expression *every musician* does not license NPIs or downward entailments in the VP.

Consider what happens when we consider a subset $X'$ of $X$ (e.g., the set of cellists). *Every $X Y$* means that $X$ 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 follows. Let the circle drawn with a solid line represent the set of snorers, and let the dashed line represent the musicians. Assume it is true that all musicians snore, so that the dashed circle is fully contained by the solid circle. Now let the dotted circle represent the set of cellists. This will be fully contained by the dashed line, because every cellist is a musician.

So, if *Every musician snores* is true, then *Every cellist snores* is also true. Since *every $X Y$* entails *every $X' Y$* for every $X'$ that is a subset of $X$, we can say that *every* is **LEFT DOWNWARD MONOTONE** (“left” because it has to do with the element on the left, $X$, rather than the element on the right, $Y$.) In general, a determiner $\delta$ is left downward monotone if $\delta X Y$ entails $\delta X' Y$ for all $X'$ that are subsets of $X$.

A determiner $\delta$ is **RIGHT DOWNWARD MONOTONE** if $\delta X Y$ entails $\delta X Y'$ for any $Y'$ that is a subset of $Y$. Let us consider whether *every* is right downward monotone. Suppose that *every $X Y$* is true. Then $X$ is a subset of $Y$. Now we will take a subset of $Y$, $Y'$. Are we guaranteed that $X$ is a subset of $Y'$? No! Consider the following scenario.
Or think about it this way: Just because every musician snores doesn’t mean that every musician snores loudly. So every is not right downward monotone.

Now let us consider some. With some, we are not guaranteed that the sentence will remain true when we replace $X$ with a subset $X'$. Some $XY$ means that 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:

![Diagram]

So, for example, suppose that Some musician snores is true. This does not mean 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 12.** For each of the examples in (2), (3), and (4), check whether Ladusaw’s generalization works. Are downward entailments licensed in exactly the places where NPIs are licensed? (Note that the examples that you need to construct in order to test this need not contain NPIs; they can be examples like the ones in (15), (16) and (17).)
Note that although Ladusaw’s generalization works surprisingly well considering its simplicity, there are certain complications. One issue is how to explain the presence of negative polarity items in questions; cf. *Did you have any problems?* On the basis of this and other data, Giannakidou (1999) offers a theory of negative polarity item licensing based on a notion called ‘veridicality’. Another issue is that presupposition needs to be taken into consideration. For example, recall that *before* and *if* license NPIs and *after* and *when* do not. So *before* and *if* should be downward-entailing and *after* and *when* should not be. Based on the following examples, it seems that *after* and *when* are not downward-entailing, and *if* is downward-entailing, but *before* is not, although intuitions are not entirely clear on this point.

(18) a. John will replace the money before he gets to France.
   b. $\not\Rightarrow$ John will replace the money before he gets to Paris.

(19) a. John will replace the money if he gets to France.
   b. $\Rightarrow$ John will replace the money if he gets to Paris.

(20) a. John will replace the money after he gets to France.
   b. $\not\Rightarrow$ John will replace the money after he gets to Paris.

(21) a. John will replace the money when he gets to France.
   b. $\not\Rightarrow$ John will replace the money when he gets to Paris.

Whether or not one gets the intuition that (18a) entails (18b) seems to depend on whether or not one assumes that John will get to Paris. Assuming that John will get to Paris, (18a) does seem to imply (18b), because John will get to France either before he gets to Paris or simultaneously.

---

3 Similarly, Condoravdi (2010) observes that *Ed left before we were in the room* does not intuitively imply *Ed left before we were in the room standing by the window*, but the inference does go through if we assume that we were standing by the window.
motivated by von Fintel (1999) to introduce the concept of *Strawson Downward-Entailment* (although von Fintel did not address *before* per se). An environment is Strawson downward-entailing if it is downward-entailing under the assumption that all of the presuppositions of both sentences are true. The discussion about how to account for the distribution and meaning of negative polarity items is a rich one that is still ongoing.


3  First order logic

This chapter introduces first order logic, which provides tools for explaining with satisfactory precision why, for example, the following argument is valid:

\[
\begin{align*}
\text{All men are mortal.} & \quad \text{(Premise 1)} \\
\text{Socrates is a man.} & \quad \text{(Premise 2)} \\
\text{Therefore, Socrates is mortal.} & \quad \text{(Conclusion)}
\end{align*}
\]

The first premise of this argument, All men are mortal, can be represented in first-order logic as follows:

\[\forall x [\text{Man}(x) \rightarrow \text{Mortal}(x)] \]

This can be read, ‘for all \( x \), if \( x \) is a man, then \( x \) is mortal’. The second premise, Socrates is a man, can be written as follows:

\[\text{Man}(s)\]

where \( s \) is a so-called INDIVIDUAL CONSTANT referring to Socrates. The conclusion, Socrates is mortal, can be written:

\[\text{Mortal}(s)\]

The semantics of first-order logic is defined in such a way that any situation (or world, or model) in which the premises hold is also a situation in which the conclusion holds. Hence, the argument is valid in this logic. (And that seems to be a point in favor of this logic as a way of representing the meaning of the corresponding argument in English, as that argument does indeed appear to be valid, according to our intuitions.)
3.1 Atomic formulas

3.1.1 Individual constants

Individual objects are named by individual constants, like $s$ above, standing for Socrates. Because they refer to individuals (rather than denoting a set or a relation or a truth value), individual constants are terms. 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,

$$s\_a\_n\_d\_j$$

is a valid individual constant, but

$$S$$

is not, nor is

$$s\ and\ j.$$

Expressions of first-order logic are interpreted relative to a particular world or scenario, called a model. The model is associated with a set of objects, those objects that exist in the model. This set of objects is called the domain of the model. An individual constant denotes a particular individual object in the domain. Unlike names in English such as Mary, individual constants are not ambiguous; they pick out exactly one object. However, note that not every individual object in the model must have a name. It is important to distinguish between the objects themselves, which are not part of the formal language, and the names for those objects, which are. For example, we might have a model whose domain consists of Bart, Lisa, Maggie, Marge, and Homer Simpson. It could nevertheless be the case that the only constants in the language are bart and lisa, corresponding to Bart Simpson and Lisa Simpson, respectively. When we write “Bart Simpson” using normal font, we are referring to the individual Bart Simpson in our
meta-language. When we write bart, we are mentioning the individual constant that is part of the formal logical language that we are defining. Since we are talking about first-order logic, first-order logic is our object language for the moment.

3.1.2 Predicate symbols and atomic formulas

Along with individual constants, first-order logic has predicate symbols, like Mortal above. Mortal is an example of a unary predicate, one which takes only one argument. Something like OlderThan would be a binary predicate, relating two individuals. The arity of a predicate is the number of arguments that it takes. 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. An example of a ternary predicate might be BETWEEN, which would hold of three objects $x, y$ and $z$, if $x$ is between $y$ and $z$.

Predicate symbols combine with the appropriate number of arguments to form atomic formulas. For example,

$$\text{Mortal}(s)$$

is an atomic formula, expressing the fact that Socrates is mortal.

$$\text{OlderThan}(s, s)$$

is also an atomic formula, expressing the absurd idea that Socrates is older than himself. That sentence will presumably always be false. In general:

Given any predicate $\pi$, if $n$ is the arity of $\pi$, and $\alpha_1, ..., \alpha_n$ is a sequence of terms, then

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

is an atomic formula.
Atomic formulas express claims that can be either true or false. Whether or not they are true or false depends on the denotations of the terms and predicates involved. Recall from above that an individual constant denotes an individual in the domain of the model. A unary predicate denotes a set of objects in the model. For example, Mortal denotes the set of individuals that are mortal according to the model. The sentence Mortal(s) is defined to be true relative to a given model just in case whatever s denotes in the model is in the set denoted by Mortal.

Let \([s]^M\) denote the semantic value of s with respect to model M, and let \([\text{Mortal}]^M\) denote the semantic value of Mortal with respect to model M. In general:

\[
[\alpha]^M = \begin{cases} 
1 & \text{if } [\alpha]^M \in [\text{Mortal}]^M, \\
0 & \text{otherwise.}
\end{cases}
\]

In general, if \(\pi\) is a unary predicate and \(\alpha\) is a term, then:

\[
[\pi(\alpha)]^M = \begin{cases} 
1 & \text{if } [\alpha]^M \in [\pi]^M, \\
0 & \text{otherwise.}
\end{cases}
\]

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 \(M\),
First order logic

and 0 otherwise.” To put it somewhat more elegantly: The predication of $\pi$ upon $\alpha$ is true if and only if the denotation of $\alpha$ is a member of the set denoted by $\pi$.

A binary predicate denotes a binary relation. For example, we might define a binary predicate $\text{OlderThan}$ which denotes the set of ordered pairs $\langle x, y \rangle$ such that $x$ is older than $y$. A predication of a binary predicate upon its two arguments is true if and only if the ordered pair consisting of the denotations of those two arguments is a member of the relation denoted by the binary predicate. For instance,

$$\text{OlderThan}(\text{bart}, \text{lisa})$$

is defined to true with respect to $M$ if and only if $\langle [\text{bart}]^M, [\text{lisa}]^M \rangle$ is a member of $[\text{OlderThan}]^M$. In general, if $\pi$ is a binary predicate and $\alpha$ and $\beta$ are terms, then:

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

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

If $\pi$ is a predicate of arity $n$ and $\alpha_1, ..., \alpha_n$ is a sequence of terms, then:

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

When defining the semantics for a first-order language, we must specify how a binary predicate like $\text{OlderThan}$ gets its value. How do we know what $[\text{OlderThan}]^M$ is? The answer is that it is determined by $M$. A model for first order logic determines not only a domain of individuals $D$, but also an interpretation function $I$, which specifies the value for all of the constants in the language. Formally, then, a model $M$ can be defined as an ordered pair $\langle D, I \rangle$, where $D$ is the domain of individuals and $I$ is the interpretation function, which, again, assigns a semantic value to every constant symbol in the language. The constant symbols
(or just “constants”) include individual constants as well as predicate symbols and function symbols (which will be introduced below). For example, the value that $I$ assigns to the constant $s$ might be Socrates, in which case we could write:

$$I(s) = \text{Socrates}$$

It is worth reiterating the distinction between the individuals in the model and the names for them. 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 above] the [philosopher Socrates himself] but rather have let [him] be represented by [his] 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 Socrates to come and occupy the right-hand side of the equation above, there would have been less of a risk of confusing the man himself with his individual constant. But as things stand, we must make do using his name in ordinary English to refer to him.

The value that $I$ assigns to the constant Philosopher might be the set containing Socrates and Plato (a rather restrictive interpretation of the term, but one that saves a lot of typing):

$$I(\text{Philosopher}) = \{\text{Socrates}, \text{Plato}\}$$
Similarly, a binary predicate would be assigned a binary relation (a set of ordered pairs) as its semantic value. In general:

If \( \alpha \) is any constant, and \( M = (D, I) \), then

\[
[\alpha]^M = I(\alpha).
\]

**Exercise 1.** Suppose we have a particular model \( M_1 = (D_1, I_1) \). Let \( D_1 = \{\text{Maggie Simpson, Bart Simpson, Homer Simpson}\} \). Suppose that in \( M_1 \), everybody loves themselves and nobody loves anybody else, and the binary predicate Loves denotes the love relation. What is then the value of \( I_1(\text{Loves})? \) Specify the relation as a set of ordered pairs.

### 3.1.3 Function symbols

Recall that a **term** is an expression that denotes an individual in the domain. So far, the only kind of term that we have seen are individual constants. But it is also possible to form syntactically complex terms using **function symbols**. An example of a function symbol would be mother, which, combined with the name of an individual, denotes the mother of that individual. For example,

\[
\text{mother}(s)
\]

could be used to denote the mother of Socrates. It does not say something about Socrates, 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.

Predicate symbols and function symbols are easy to confuse with each other, because they both take arguments in parenthe-
ses. To distinguish between them, we will establish a convention whereby predicate symbols start with uppercase letters, and function symbols start with lowercase letters. This way, all terms, both individual constants and complex terms formed with function symbols, start with lowercase letters. Any sequence of numbers or letters or underscores may follow the initial letter, but no spaces.

Functions, like predicates, are associated with a particular arity. The mother function has arity 1 (i.e., it is a **UNARY FUNCTION**). An example of a function with arity 2 might be eldestChild. Something like:

```
eldestChild(john, mary)
```

could be used to denote the eldest child of John and Mary. In general:

Given any function $\gamma$, if $n$ is the arity of $\gamma$, then:

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

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

Semantically, function symbols combine with their arguments in a slightly different manner from the way predicates do. Functions symbols denote functions (of the kind discussed in the previous chapter, namely, relations of a certain type), and 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 $M$, mother denotes a function that gives Phaenarete when given the individual Socrates as an argument. Then $\text{mother}(s)$ denotes Phaenarete, i.e., $\llbracket \text{mother}(s) \rrbracket^M = \text{Phaenarete}$. In general:
If $\gamma$ is a unary function, and $\alpha$ is a term, then:

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

This 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, in a realistic model according to which phaenarete denotes Phaenarete and sophroniscus denotes Sophroniscus, the following expression:

```
eldestChild(phaenarete,sophroniscus)
```

would denote the result of applying the function denoted by eldestChild to the ordered pair (Phaenarete,Sophroniscus). In general:

If $\gamma$ is a binary function, and $\alpha$ and $\beta$ are terms, then:

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

This definition can be generalized to accommodate functions of arbitrary arity:
If \( \gamma \) 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((\[ \alpha_1 \]_M, \ldots, \[ \alpha_n \]_M))
\]

So much for the internal parts of atomic formulas. In the next section, we will start joining atomic formulas together into complex formulas.

### 3.1.4 Boolean connectives

**Exercise 2.** Some of the following are fallacies; some are valid argument forms. Based on your intuition, decide which are which.

(a) **Affirming the consequent**
   Premise 1: if \( P \) then \( Q \)
   Premise 2: \( Q \)
   Conclusion: \( P \)

(b) **Reductio ad absurdum**
   Premise 1: not \( P \) implies \( Q \)
   Premise 2: not \( P \) implies not \( Q \)
   Conclusion: \( P \)

(c) **Hypothetical syllogism**
   Premise 1: if \( P \) then \( Q \)
   Premise 2: if \( Q \) then \( R \)
   Conclusion: if \( P \) then \( R \)

(d) **Modus Tollens**
   Premise 1: if \( P \) then \( Q \)
   Premise 2: not \( Q \)
   Conclusion: not \( P \)
(e) **Proof by cases**
Premise 1: P or Q
Premise 2: if P then R
Premise 3: if Q then R
Conclusion: R

(f) **Affirming a disjunct**
Premise 1: P or Q
Premise 2: P
Conclusion: not Q

(g) **Denying the antecedent**
Premise 1: if P then Q
Premise 2: not P
Conclusion: not Q

(h) **Disjunctive syllogism**
Premise 1: P or Q
Premise 2: not Q
Conclusion: P

(i) **De Morgan (negation of a conjunction)**
Premise: not (P and Q)
Conclusion: not P or not Q

(j) **De Morgan (negation of a disjunction)**
Premise: not (P or Q)
Conclusion: not P and not Q

(k) **Constructive dilemma**
Premise 1: if P then Q
Premise 2: if R then S
Premise 3: P or R
Conclusion: Q or S

(l) **Material implication**
Premise 1: P implies Q
Conclusion: not P or Q
(m) **Illicit major**

Premise 1: P implies Q

Premise 2: R implies not P

Conclusion: R implies not Q

Suppose you ask me if I’m free today or tomorrow and I say no. That means that I’m not free today, and I’m not free tomorrow. So in general:

\[
\neg (P \lor Q)
\]

means

\[
\neg P, \text{ and } \neg Q
\]

On the other hand, suppose you ask me if I’m free today *and* tomorrow and I say no. That is not quite as strong: it means that either I’m not free today or I’m not free tomorrow (or both). Thus

\[
\neg (P \land Q)
\]

means

\[
\neg P, \text{ or } \neg Q
\]

These are **De Morgan’s laws**. By specifying an interpretation for *and*, *or*, and *not*, we can capture the logical relationships between these sentences.

In first-order logic, atomic formulas can be combined together to form larger formulas using **connectives**. For example:

\[
[\text{Happy}(a) \land \text{Happy}(b)]
\]

can be read ‘a is happy and b is happy’. The \(\land\) symbol represents ‘and’. In general, if \(\phi\) is a formula and \(\psi\) is a formula, then \([\phi \land \psi]\) is also a formula, known as the **conjunction** of \(\phi\) and \(\psi\).
The **disjunction** of \( \phi \) and \( \psi \) is written \( [\phi \lor \psi] \). For example:

\[
[\text{Happy}(a) \lor \text{Happy}(b)]
\]
can be read ‘a is happy or b is happy’. The word ‘or’ in English sometimes has an **inclusive** interpretation, and sometimes has an **exclusive** interpretation. A sentence like *I’ll visit him today or tomorrow* typically implies that the speaker will visit the person on one of those days but not both. But with a sentence like *If John volunteers Monday or Wednesday, then we have all the help we need*, it is less likely that an exclusive interpretation is intended, suggesting that there won’t be enough help if he volunteers on both days. In this situation, it is natural to interpret *or* in an inclusive manner, where the condition is satisfied even when both disjuncts are true. (A **disjunct** is a phrase or sentence that is joined to another one by *or*.)

Let’s express the rule using symbols to bring out its structure.

\[
P = \text{John volunteers on Monday}
\]
\[
Q = \text{John volunteers on Wednesday}.
\]

Let us represent “John volunteers on Monday or Wednesday” as follows:

\[
P \lor Q
\]

Now, this \( \lor \) symbol should be interpreted in such a way that \( P \lor Q \) is true whenever at least one of \( P \) or \( Q \) is true. There are four possibilities: \( P \) is true and \( Q \) is false; \( P \) is false and \( Q \) is true; both are true; and both are false. Only in the last case should the disjunction be seen as false. This interpretation of \( \lor \) can be represented using a **truth table**, as follows.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( P \lor Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
We use the number 1 to represent “true” and the number 0 to represent “false”. This table interprets $\lor$ as **inclusive disjunction**, because the statement is true even in the case where *both* of the disjuncts are true.

**Exclusive disjunction** specifies that only one of the disjuncts is true:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \oplus Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 negation: If I say *I won’t see you today or tomorrow*, it means that I will not see you today and I won’t see you tomorrow. We can get this interpretation by negating the inclusive interpretation (‘it is not the case that at least one of the disjuncts is true’). Conversational implicatures of the kind that would be involved here (‘scalar implicatures’) typically disappear under negation, so this fact is easily explained under the implicature analysis. Under the ambiguity analysis, it is not clear why an exclusive reading should disappear under negation. (There is of course much more to say on this issue.)

If I say, *I’ll see you today and tomorrow*, then my statement is true if and only if I’ll see you today and I’ll see you tomorrow. This is called a **conjunction**. Conjunction can be represented as follows:

$$P \land Q$$

The truth table for conjunction is as follows:
So far, we have discussed three different **BINARY CONNECTIVES**: inclusive disjunction, exclusive disjunction, and conjunction. There is one **UNARY CONNECTIVE** in the language, for negation, written \( \neg \). Because negation is a unary connective, the truth table for negation is simple. If \( P \) is true, then \( \neg P \) is false. If \( P \) is false, then \( \neg P \) is true:

<table>
<thead>
<tr>
<th>( P )</th>
<th>( \neg P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Let us consider when the negation of \( P \land Q \) is true. To find out, we first find out when \( P \land Q \) is true, and then apply negation to that.

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</th>
<th>( P \land Q )</th>
<th>( \neg[P \land Q] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: We are using the brackets [ ] to show that we are applying negation to the conjunction of \( P \) and \( Q \), rather than \( P \). Conjunction, disjunction, and negation make up the so-called **BOOLEAN CONNECTIVES**, named after the logician George Boole. Whenever you do a “Boolean search”, you are using these connectives.

### 3.1.5 Conditionals

All of the connectives we have seen are **TRUTH-FUNCTIONAL**, in the sense that the truth of formulas containing them depends on
the truth values of the formulas it joins together. There is another truth-functional connective that corresponds roughly to “if... then” in natural language, called \textbf{MATERIAL IMPLICATION}, and written $\rightarrow$. A formula of the form $P \rightarrow Q$ is false only when $P$ is true and $Q$ is false, and true otherwise. The truth table looks like this:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \rightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Indeed, suppose I make the following claim: “If it’s sunny, then there are people sitting outside.” This claim would only be falsified by a scenario in which it is sunny, but there are no people sitting outside. It would not be falsified by a scenario in which it’s sunny and there are people sitting outside, nor would it be falsified by any scenario in which it’s not sunny. What I’ve said does not bear on what happens when it’s not sunny.

Another motivation for treating material implication as the meaning of ‘if... then...’ is the fact that this treatment explains why certain inferences involving ‘if... then...’ are valid, and others are not. For example, it explains Modus Ponens:

\begin{align*}
\text{If } P \text{ then } Q & \quad \text{(Premise 1)} \\
\; \quad P & \quad \text{(Premise 2)} \\
\text{Therefore, } Q & \quad \text{(Conclusion)}
\end{align*}

(A particular instance: If it’s sunny then there are people sitting outside. It’s sunny. Therefore, there are people sitting outside.) An argument is valid if the conclusion is true whenever the conjunction of the premises is true. According to the truth table for material implication, $P \rightarrow Q$ and $P$ are jointly true whenever both $P$ and $Q$ are true. Therefore, given this treatment of ‘if... then”, the conclusion, $Q$, is true whenever the premises are both true.
However, 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, conditionals 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 need somewhat more sophisticated technology. (The interested reader is referred to the work of David Lewis, Robert Stalnaker, and Angelika Kratzer, among many others.) But if we are forced to identify the truth-functional connective that most closely resembles *if then*, we will have to choose material implication.

### 3.2 Equivalence and tautology

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:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\neg P$</th>
<th>$\neg\neg P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

This is a general way to determine whether two formulas are equivalent: Construct a truth table with columns for both expressions, and check whether the pattern of 1s and 0s matches. If so, then the two expressions are equivalent.
Exercise 3. Using truth tables, check whether the following pairs of formulas are equivalent.

(a) \([P \lor Q]; \neg[P \land \neg Q]\)
(b) \([P \rightarrow Q]; [\neg P \lor Q]\)
(c) \([P \land Q]; \neg P \lor \neg Q\)
(d) \([P \lor \neg Q]; \neg P \land \neg Q\)
(e) \([P \rightarrow Q]; [\neg Q \rightarrow \neg P]\)
(f) \([P \rightarrow P]; [P \lor \neg P]\)

(Note that the truth table for this one should only contain two rows, since it doesn’t mention Q.)

Two logical expressions are **contradictory** if for every assignment of values to their variables, their truth values are different. For example \(P\) and \(\neg P\) are contradictory.

\[
\begin{array}{c|c}
P & \neg P \\
\hline
1 & 0 \\
0 & 1 \\
\end{array}
\]

Another contradictory pair is \(P \rightarrow Q\) and \(P \land \neg Q\).

\[
\begin{array}{c|c|c|c|c}
P & Q & P \rightarrow Q & \neg Q & P \land \neg Q \\
\hline
1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
\end{array}
\]

A **tautology** is an expression that is true in every situation. So you can tell whether an expression is a tautology by looking at
the pattern of 1s and 0s in the column underneath it in a truth table: If they’re all true, then it is a tautology. Here is a tautology: \( P \lor \neg P \) (e.g. It is raining or it is not raining):

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>\neg P</th>
<th>P \lor \neg P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Fact: 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:

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>\neg P</th>
<th>\neg \neg P</th>
<th>\neg P \leftrightarrow \neg \neg P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Exercise 4. Which of the following are tautologies?

(a) \( P \lor Q \)

(b) \([ P \rightarrow Q ] \lor [ Q \rightarrow P ]\)

(c) \([ P \rightarrow Q ] \leftrightarrow [ \neg Q \lor \neg P ]\)

(d) \([ [ P \lor Q ] \rightarrow R ] \leftrightarrow [ [ P \rightarrow Q ] \land [ P \rightarrow Q ] ]\)

Support your answer with truth tables.

3.3 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 syntax 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 languages have both grammar and semantics. The grammar specifies the well-formed formulas of the language. The semantics specifies the semantic value of every well-formed formula, given a model.

### 3.3.1 Syntax of $L_0$

1. **Basic Expressions**
   - Individual constants: $d$, $n$, $j$, $m$
   - Function symbols
     - Unary: mother, father
     - Binary: eldestChild
   - Predicate symbols
     - Unary: Happy, Bored, Linguist, Philosopher, Cel-list, Smokes
     - Binary: Kissed, Loves, Admires, Siblings

2. **Terms**
   - Every individual constant is a term.
   - If $\pi$ is a function symbol of arity $n$, and $\alpha_1, ..., \alpha_n$ are terms, then $\pi(\alpha_1, ..., \alpha_n)$ is a term.$^1$

3. **Atomic formulas**
   - If $\pi$ is a function of arity $n$ and $\alpha_1, ...\alpha_n$ are terms, then $\pi(\alpha_1, ...\alpha_n)$ is an atomic formula.$^2$

---

$^1$Special cases:
- If $\pi$ is a unary function symbol and $\alpha$ is a term then $\pi(\alpha)$ is a term.
- If $\pi$ is a binary function symbol and $\alpha$ and $\beta$ are terms then $\pi(\alpha, \beta)$ is a term.

$^2$Special cases:
• If α and β are terms, then α = β is an atomic formula.

4. Negation

• If φ is a formula, then ¬φ is a formula.

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

• \([ φ \land ψ ]\)

  ‘φ and ψ’

• \([ φ \lor ψ ]\)

  ‘φ or ψ’

• \([ φ → ψ ]\)

  ‘if φ then ψ’

• \([ φ ↔ ψ ]\)

  ‘φ if and only if ψ’

Note that the outer square brackets with binary connectives and quantifiers 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 L₀

The denotation of an expression α relative to a model M is written \([α]_M\). A model \(M = (D, I)\) determines a domain of individuals \(D\) and an interpretation function \(I\).

1. Basic expressions

• If α is a constant, then \([α]_M = I(α)\).

2. Complex terms

• If π is a function of arity \(n\), and \(α_1, ..., α_n\) is a sequence of \(n\) terms, then \([π(α_1, ..., α_n)]_M = [π]_M([α_1]_M, ..., [α_n]_M)\)

\[3\]

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

\[3\] Special cases:
3. Atomic formulas

- **Predication**: If $\pi$ is a predicate of arity $n$, and $\alpha_1, ..., \alpha_n$ is a sequence of $n$ terms, then $\llbracket \pi(\alpha_1, ..., \alpha_n) \rrbracket^M = 1$ if $\langle \llbracket \alpha_1 \rrbracket^M, ..., \llbracket \alpha_n \rrbracket^M \rangle \in \llbracket \pi \rrbracket$, and 0 otherwise.\(^4\)

- **Equality**: If $\alpha$ and $\beta$ are terms, then $\llbracket \alpha = \beta \rrbracket^M = 1$ if $\llbracket \alpha \rrbracket^M = \llbracket \beta \rrbracket^M$, and 0 otherwise.

4. Negation

- $\llbracket \neg \phi \rrbracket^M = 1$ if $\llbracket \phi \rrbracket^M = 0$, and 0 otherwise.

5. Binary connectives

- $\llbracket \phi \land \psi \rrbracket^M = 1$ if $\llbracket \phi \rrbracket^M = 1$ and $\llbracket \psi \rrbracket^M = 1$, and 0 otherwise.

- $\llbracket \phi \lor \psi \rrbracket^M = 1$ if $\llbracket \phi \rrbracket^M = 1$ or $\llbracket \psi \rrbracket^M = 1$, and 0 otherwise.

- $\llbracket \phi \rightarrow \psi \rrbracket^M = 1$ if $\llbracket \phi \rrbracket^M = 0$ or $\llbracket \psi \rrbracket^M = 1$, and 0 otherwise.

- $\llbracket \phi \leftrightarrow \psi \rrbracket^M = 1$ if $\llbracket \phi \rrbracket^M = \llbracket \psi \rrbracket^M$, and 0 otherwise.

3.4 Quantification

Consider the sentence *Beth owns a truck*. Construing ownership as a binary relation that holds between potential owners and the objects they own, this sentence is true in any model where Beth stands in the ownership relation to an object which is a truck. How can we express this formally? If we had names for all of the trucks

\(^4\)Special cases:

- When $\pi$ is a function of arity 1, then $\llbracket \pi(\alpha) \rrbracket = \llbracket \pi \rrbracket^M(\llbracket \alpha \rrbracket^M)$.
- When $\pi$ is a function of arity 2, then $\llbracket \pi(\alpha, \beta) \rrbracket = \llbracket \pi \rrbracket^M(\llbracket \alpha \rrbracket^M, \llbracket \beta \rrbracket^M)$.
in the model, then we could express this using the tools we have by saying something along the lines of, “Beth owns Truck 1 or Beth owns Truck 2 or ...” and so on for all of the trucks. But this is quite inconvenient. We don’t want to have to name all the trucks. All we want to say is that there is some object, call it \( x \), such that Beth owns \( x \) and \( x \) is a truck. This can be done using variables. The condition that the object should satisfy may be written as follows:

\[
[\text{Owns} (\text{beth}, x) \land \text{Truck} (x)]
\]

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 \text{BOUND} by any quantifier (so it is \text{FREE}). To make the claim that there is some object \( x \) that satisfies this condition, we may use the \text{EXISTENTIAL QUANTIFIER}, \( \exists \).

\[\exists x [\text{Owns}(\text{beth}, x) \land \text{Truck}(x)]\]

This can be read, “There exists an \( x \) such that Beth owns \( x \) and \( x \) is a truck.” And this formula will be true in any model where there is a truck that Beth owns.

The other quantifier of first-order logic is the \text{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 \textit{everything} satisfies the condition. Thus Beth owns everything and everything is a truck. 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 \textit{Every athlete got an A}. We might represent this as follows:

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

This can be read, “For all \( x \), if \( x \) is an athlete, then the grade \( x \) received was an A.” Note that we would be saying something very
different if we had a conjunction symbol ($\land$) instead of a material implication arrow ($\rightarrow$) in this formula, thus:

$$\forall x[\text{Athlete}(x) \land \text{ReceivedGrade}(x, a)]$$

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.”

Now consider the following formula:

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

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 notice that “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:

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

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:

$$\forall x. \neg \text{Happy}(x)$$
\[ \neg \forall x. \text{Happy}(x) \]

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 5.** 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) \[ \forall x. \text{Bored}(x) \]

(b) \[ \forall x. [\text{Bored}(x) \land \text{Happy}(x)] \]

(c) \[ \exists x. [\text{Bored}(x) \land \text{Happy}(x)] \]

(d) \[ \exists x. [\text{Bored}(x) \lor \text{Happy}(x)] \]

(e) \[ \forall x. [\text{Bored}(x) \rightarrow \text{Happy}(x)] \]

(f) \[ \forall x. \neg \text{Bored}(x) \]

(g) \[ \exists x. \neg \text{Bored}(x) \]

(h) \[ \neg \exists x. \text{Bored}(x) \]

(i) \[ \forall x. \exists y. \text{Loves}(y)(x) \]

**Exercise 6.** 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 bored and happy.
(b) Everybody is bored and happy.
(c) Everybody who is bored is happy.
(d) Nobody is bored.
(e) Somebody is not bored.
(f) Somebody is bored or happy.
(g) Everybody loves somebody.
(h) Somebody loves everybody.

**Exercise 7.** Which of the following statements in first-order logic better represents the meaning of *Every cellist smokes*?

(a) $\forall x. [\text{Cellist}(x) \rightarrow \text{Smokes}(x)]$
(b) $\forall x. [\text{Cellist}(x) \land \text{Smokes}(x)]$

**Exercise 8.** 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 cats.

Feel free to add as many logical constants as you need.
Exercise 9. 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 person 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 logical constants as you need.

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 $\exists$. Given any variable $u$, if $\phi$ is a formula, then

$$[\forall u. \phi]$$
is a formula, and so is

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

For example, $$\forall x. \text{Happy}(x)$$ 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. $$\forall x[\text{Happy}(x) \rightarrow \text{Bored}(x)].$$ In a formula of the form $$\forall u \phi$$ or $$\exists u \phi,$$ $$\phi$$ is called the *scope* of the quantifier. This term sometimes comes in handy.

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 $$(D, I)$$ 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,

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

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

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

is true. Likewise, informally,

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

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

The formula $$\text{Happy}(x)$$ does not make a claim that could be true or false relative to any given model. In order to decide whether $$\text{Happy}(x)$$ is true, we need to know not only about the model, but also how to interpret $$x.$$ If $$x$$ is interpreted as someone who is happy, then the formula is true; otherwise not. The device that we will use to interpret variables is called an assignment function. 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:
First order logic

\[ g_1 = \begin{bmatrix} 
  x & \rightarrow & \text{Maggie} \\
  y & \rightarrow & \text{Bart} \\
  z & \rightarrow & \text{Bart} \\
  \ldots 
\end{bmatrix} \quad g_2 = \begin{bmatrix} 
  x & \rightarrow & \text{Bart} \\
  y & \rightarrow & \text{Homer} \\
  z & \rightarrow & \text{Bart} \\
  \ldots 
\end{bmatrix} \]

The domain of an assignment function is the set of variables \( \nu_n \) for all \( n \).

In order to interpret an expression, then, we need both a model and an assignment function. We typically use the letter \( g \) to stand for an assignment function, so instead of \( \llbracket \phi \rrbracket^M \) we will now write:

\( \llbracket \phi \rrbracket^{M,g} \)

where \( g \) stands for an assignment function. The denotation of the variable \( x \) with respect to model \( M \) and assignment function \( g \), written:

\( \llbracket x \rrbracket^{M,g} \)

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

(1) \( \llbracket x \rrbracket^{M,g} = g(x) \)

For example, \( \llbracket x \rrbracket^{M,g_1} = g_1(x) = \text{Maggie} \), and \( \llbracket x \rrbracket^{M,g_2} = g_2(x) = \text{Bart} \) for any model \( M \).

**Exercise 10.**

(a) What is \( g_1(y) \)?

(b) What is \( \llbracket y \rrbracket^{M,g_1} \) (for any model \( M \))? 

(c) What is \( g_2(y) \)?

(d) What is \( \llbracket y \rrbracket^{M,g_2} \) (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, \( \llbracket \text{Happy} \rrbracket^{M_1,g_1} \) will be the same as \( \llbracket \text{Happy} \rrbracket^{M_1,g_2} \) for any two assignments \( g_1 \) and \( g_2 \), because Happy is a constant. Its value depends only on the model. But the value of the formula

\[
\text{Happy}(x)
\]

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

Now let us consider the formula \( \exists x. \text{Happy}(x) \). This is true if we can find one individual to assign \( x \) to such that \( \text{Happy}(x) \) is true. Suppose we are trying to determine whether \( \exists x. \text{Happy}(x) \) 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 \( g \) save that \( g(u) = k \). If \( g \) already maps \( u \) to \( k \) then, \( g[u \mapsto k] \) is the same as \( g \). This lets us keep everything the same in \( g \) except for the variable of interest. For example, using \( g_1 \) from above,

\[
g_1 = \left[ \begin{array}{c} x \rightarrow \text{Maggie} \\ y \rightarrow \text{Bart} \\ z \rightarrow \text{Bart} \\ \cdots \end{array} \right]
\]

\( g_1[y \mapsto \text{Homer}] \) would be as follows:

\[
g_1[y \mapsto \text{Homer}] = \left[ \begin{array}{c} x \rightarrow \text{Maggie} \\ y \rightarrow \text{Homer} \\ z \rightarrow \text{Bart} \\ \cdots \end{array} \right]
\]
We changed it so that $y$ maps to Homer and kept everything else the same.

**Exercise 11.**

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

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

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

With this terminology, we can say:

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

What this says is that given a model $M$ and an assignment function $g$, the sentence $\exists x. \text{Happy}(x)$ 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 $\text{Happy}(x)$ true.

Now, if we wanted to show that the formula $\forall x. \text{Happy}(x)$ 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 $\text{Happy}(x)$ turns out to be true no matter what the assignment function maps $x$ to, then $\forall x. \text{Happy}(x)$ is true. Otherwise it is false. So the official semantics of the universal quantifier is as follows:

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

### 3.4.1 Syntax of $L_1$

Let us now summarize the syntactic rules of our language.
1. **Basic Expressions**

- Individual constants: d, n, j, m
- Individual variables: $v_n$ for every natural number $n$
- Function symbols
  - Unary: mother, father
  - Binary: eldestChild
- Predicate symbols
  - Unary: Happy, Bored, Linguist, Philosopher, Cellist, Smokes
  - Binary: Kissed, Loves, Admires, Siblings

2. **Terms**

- Every individual constant is a term.
- Every individual variable is a term.
- If $\pi$ is a function symbol of arity $n$, and $\alpha_1, ..., \alpha_n$ are terms, then $\pi(\alpha_1, ..., \alpha_n)$ is a term.\(^5\)

3. **Atomic formulas**

- **Predication**
  If $\pi$ is a function of arity $n$ and $\alpha_1, ..., \alpha_n$ are terms, then $\pi(\alpha_1, ..., \alpha_n)$ is an atomic formula.\(^6\)

\(^5\)Special cases:
- If $\pi$ is a unary function symbol and $\alpha$ is a term then $\pi(\alpha)$ is a term.
- If $\pi$ is a binary function symbol and $\alpha$ and $\beta$ are terms then $\pi(\alpha, \beta)$ is a term.

\(^6\)Special cases:
- If $\pi$ is a unary predicate and $\alpha$ is a term, then $\pi(\alpha)$ is a formula.
- If $\pi$ is a binary predicate and $\alpha$ and $\beta$ are terms, then $\pi(\alpha, \beta)$ is formula.
• **Equality**  
  If $\alpha$ and $\beta$ are terms, then $\alpha = \beta$ is an atomic formula.

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

5. **Binary connectives**  
  If $\phi$ is a formula and $\psi$ is a formula, then so are:
  
  • $[\phi \land \psi]$  
  ‘$\phi$ and $\psi$’
  
  • $[\phi \lor \psi]$  
  ‘$\phi$ or $\psi$’
  
  • $[\phi \rightarrow \psi]$  
  ‘if $\phi$ then $\psi$’
  
  • $[\phi \leftrightarrow \psi]$  
  ‘$\phi$ if and only if $\psi$’

6. **Quantifiers**  
  If $u$ is a variable and $\phi$ is a formula, then both of the following are formulas:
  
  • $[\forall u. \phi]$  
  ‘for all $u$: $\phi$’
  
  • $[\exists u. \phi]$  
  ‘there exists a $u$ such that $\phi$’

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

• In an atomic formula, any variable is free.

• The free variables in $\phi$ are also free in $\neg \phi$, and the free variables in $\phi$ and $\psi$ are free $[\phi \land \psi]$, $[\phi \lor \psi]$, $[\phi \rightarrow \psi]$, and $[\phi \leftrightarrow \psi]$.

• All of the free variables in $\phi$ are free in $[\forall u. \phi]$ and $[\exists u. \phi]$, except for $u$, and every occurrence of $u$ in $\phi$ 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**.
closed formula is also called a **sentence**. Note that 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. For example, instead of:

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

we can write:

\[ \forall x. [\text{Linguist}(x) \rightarrow \exists y. \text{Admires}(y)(x)] \]

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, \( \forall x. [P(x) \land Q(x)] \) can be rewritten as \( \forall x. P(x) \land Q(x) \), 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 will drop the dot when we have multiple binders in a row. Thus instead of:

\[ \forall x. \exists y. \text{Admires}(x, y) \]

we can write:

\[ \forall x \exists y. \text{Admires}(x, y) \]
Note that the dot is always optional in the Lambda Calculator (on its default setting).

### 3.4.2 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 $M = (D, I)$ where $D$ is the set of individuals and $I$ is a function mapping each 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 $\alpha$ relative to $M$ and $g$, written $[\alpha]^{M,g}$, is defined as follows:

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

2. **Complex terms**
   - If $\pi$ is a function of arity $n$, and $\alpha_1, ..., \alpha_n$ is a sequence of $n$ terms, then:\footnote{Special cases:

  - When $\pi$ is a function of arity 1, then:

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

  - When $\pi$ is a function of arity 2, then:

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

\[
[\pi(\alpha_1, ..., \alpha_n)]^{M,g} = [\pi]^{M,g}([\alpha_1]^{M,g}, ..., [\alpha_n]^{M,g})
\]
3. Atomic formulas

- **Predication**
  If \( \pi \) is a predicate of arity \( n \), and \( \alpha_1, \ldots, \alpha_n \) is a sequence of \( n \) terms, then:
  \[
  \llbracket \pi(\alpha_1, \ldots, \alpha_n) \rrbracket^M,g = 1 \text{ if } \llbracket \alpha_1 \rrbracket^M,g, \ldots, \llbracket \alpha_n \rrbracket^M,g \in \llbracket \pi \rrbracket^M, \text{ and 0 otherwise.}\]

- **Equality**
  If \( \alpha \) and \( \beta \) are terms, then
  \[
  \llbracket \alpha = \beta \rrbracket^M,g = 1 \text{ if } \llbracket \alpha \rrbracket^M,g = \llbracket \beta \rrbracket^M,g, \text{ and 0 otherwise.}\]

4. Negation

- \( \llbracket \neg \phi \rrbracket^M,g = 1 \text{ if } \llbracket \phi \rrbracket^M,g = 0, \text{ and 0 otherwise.} \)

5. Binary Connectives

- \( \llbracket \phi \land \psi \rrbracket^M,g = 1 \text{ if } \llbracket \phi \rrbracket^M,g = 1 \text{ and } \llbracket \psi \rrbracket^M,g = 1, \text{ and 0 otherwise.} \)
- \( \llbracket \phi \lor \psi \rrbracket^M,g = 1 \text{ if } \llbracket \phi \rrbracket^M,g = 1 \text{ or } \llbracket \psi \rrbracket^M,g = 1, \text{ and 0 otherwise.} \)
- \( \llbracket \phi \rightarrow \psi \rrbracket^M,g = 1 \text{ if } \llbracket \phi \rrbracket^M,g = 0 \text{ or } \llbracket \psi \rrbracket^M,g = 1, \text{ and 0 otherwise.} \)
- \( \llbracket \phi \leftrightarrow \psi \rrbracket^M,g = 1 \text{ if } \llbracket \phi \rrbracket^M,g = \llbracket \psi \rrbracket^M,g, \text{ and 0 otherwise.} \)

6. Quantification

---

\( ^8 \) Special cases:
- If \( \pi \) is a unary predicate and \( \alpha \) is a term, then:
  \[
  \llbracket \pi(\alpha) \rrbracket^M = 1 \text{ if } \llbracket \alpha \rrbracket^M \in \llbracket \pi \rrbracket^M, \text{ and 0 otherwise.} \]
- If \( \pi \) is a binary predicate and \( \alpha \) and \( \beta \) are terms, then:
  \[
  \llbracket \pi(\alpha, \beta) \rrbracket^M = 1 \text{ if } \llbracket \alpha \rrbracket^M, \llbracket \beta \rrbracket^M \in \llbracket \pi \rrbracket^M, \text{ and 0 otherwise.} \]
• $\left[\forall v. \phi\right]^{M,g} = 1$ if for all individuals $k \in D$:

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

and 0 otherwise.

• $\left[\exists v. \phi\right]^{M,g} = 1$ if there is an individual $k \in D$ such that:

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

and 0 otherwise.

Note that the choice of assignment function doesn’t always make a difference for the interpretation of an expression. The choice of assignment function only makes a difference when the formula contains free variables. For example, in the formula

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

the variable $x$ is not bound by any quantifier (so it is a **free variable**). So the semantic value of this formula relative to $M$ and $g$ depends on what $g$ assigns to $x$. In contrast, a closed formula such as $\forall x. \text{Happy}(x)$ has the same value relative to every assignment function.

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:

$$\forall x. [\text{Linguist}(x) \rightarrow \exists y. [\text{Philosopher}(y) \land \text{Admires}(y)(x)]]$$

$$\exists y. [\text{Philosopher}(y) \land \forall x. [\text{Linguist}(x) \rightarrow \text{Admires}(y)(x)]]$$

We will spare the reader 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 outscope 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 12.** Is $\forall x.\operatorname{Happy}(x)$ true in $M_1$? Give a one-sentence explanation why or why not.

**Exercise 13.** Consider the following formulas.

(a) $\operatorname{Happy}(m) \land \operatorname{Happy}(m)$  
(b) $\operatorname{Happy}(k)$  
(c) $\operatorname{Happy}(m, m)$  
(d) $\neg \neg \operatorname{Happy}(n)$  
(e) $\forall x.\operatorname{Happy}(x)$  
(f) $\forall x.\operatorname{Happy}(y)$  
(g) $\exists x.\operatorname{Loves}(x, x)$  
(h) $\exists x.\exists z.\operatorname{Loves}(x, z)$  
(i) $\exists x.\operatorname{Loves}(x, z)$  
(j) $\exists x.\operatorname{Happy}(m)$

**Questions:**

(a) Which of the above are well-formed formulas of $L_1$?

(b) 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?)
**Exercise 14.** Recall our fantasy model $M_f$, where everybody is happy:

$$I_f(\text{Happy}) = \begin{bmatrix}
\text{Bart} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 1 \\
\text{Maggie} & \rightarrow & 1 
\end{bmatrix}$$

(a) What is $\llbracket x \rrbracket^{M_f,g_{\text{Bart}}}$? Apply the $L_1$ semantic interpretation rule for variables.

(b) What is $\llbracket \text{Happy} \rrbracket^{M_f,g_{\text{Bart}}}$? Apply the relevant $L_1$ semantic interpretation rule.

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

(d) Use the rule you identified in your answer to the previous question, explain carefully why $\llbracket \text{Happy}(x) \rrbracket^{M_f,g_{\text{Bart}}} = 1$.

**Exercise 15.** Let $g$ be defined such that $x \mapsto c$, $y \mapsto b$, and $z \mapsto a$, and suppose that in $M_1$, everybody loves themselves and nobody loves anybody else, and the binary predicate Loves denotes this love relation.

(a) Calculate:

(a) $\llbracket x \rrbracket^{M_1,g}$
(b) $\llbracket m \rrbracket^{M_1,g}$
(c) $\llbracket \text{Loves} \rrbracket^{M_1,g}$
(d) $\llbracket \text{Loves}(x,m) \rrbracket^{M_1,g}$
(b) List all of the value assignments that are exactly like \( g \) except possibly for the individual assigned to \( x \), and label them \( g_1 \cdots g_n \).

(c) For each of those value assignments \( g_i \) in the set \( \{g_1, \ldots, g_n\} \), calculate \( \llbracket \text{Loves}(x, m) \rrbracket^{M, g_i} \).

(d) On the basis of these and the semantic rule for universal quantification calculate \( \llbracket \forall x. \text{Loves}(x, m) \rrbracket^{M, g} \) and explain your reasoning.

**Exercise 16.** 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 \( \llbracket \text{Loves}(x, m) \rrbracket^{M, g_i} \neq \llbracket \text{Loves}(x, m) \rrbracket^{M, g_j} \).

**Exercise 17.** Determine by a detailed consideration of the relevant value considerations whether \( \forall x \forall y [\text{Kissed}(x, y) \rightarrow \text{Kissed}(y, x)] \) is true with respect to \( M_1 \). If it is false, find a model \( M_2 = \langle D_2, I_2 \rangle \) such that the sentence is true with respect to \( M_2 \).

**Exercise 18.** A sentence is *valid* if it is true in every model. Show that \( \exists x [\text{Happy}(x) \rightarrow \forall y \text{Happy}(y)] \) is valid.

**Exercise 19.** In the run-up to the 2016 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 logical constants as you need.
4 Typed lambda calculus

4.1 Lambda abstraction

In the next chapter we will define a system that translates expressions of (a defined fragment of) English into a formal logical language. In order to do this, we need to be able to assign meanings to a wide range of expressions. Consider the sentence Everybody loves Maggie, which might be translated into $L_1$ as:

$$\forall x[\text{Person}(x) \rightarrow \text{Loves}(x, m)]$$

Some parts of the sentence Everybody loves Maggie can be straightforwardly mapped into expressions in $L_1$. For example, we might say that (relative to a given model) the English name Maggie picks out a particular element of the domain, namely Maggie. So it makes sense to translate the name Maggie as an individual constant, such as $m$, as this is the sort of denotation that individual constants have. The English verb loves could be thought of as denoting a binary relation (a set of ordered pairs of individuals in the domain), the sort of thing denoted by a binary predicate. Let us therefore assume that Loves is a binary predicate and that the verb maps to it. But what does a verb phrase like loves Maggie map onto? Intuitively, it denotes a formula of the following form:

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

where the first argument of Loves is missing. A similar problem arises with expressions like Everybody. Intuitively, it denotes some-
thing like the following:

\[ \forall x. \text{Person}(x) \rightarrow \underline{}(x) \]

where \underline{} is a placeholder for some predicate. The language of typed lambda calculus, which we present now, gives us the tools to express this idea of a placeholder. Using the \( \lambda \) symbol, we can abstract over the missing piece.

The first major change is in the inventory of syntactic categories of the language. Our language \( L_0 \) had a finite set of syntactic categories: terms, unary predicates, binary predicates, and formulas. In the system we present next, we will have an infinite set of syntactic categories. These categories will be called types. The set of types is defined as follows:

- \( e \) is a type
- \( t \) is a type
- If \( \sigma \) is a type and \( \tau \) is a type, then \( \langle \sigma, \tau \rangle \) is a type.
- Nothing else is a type.

In the third clause, \( \sigma \) and \( \tau \) are variables over types, rather than particular types. This third clause implies that, for example, \( \langle e, t \rangle \) is a type, since both \( e \) and \( t \) are types. And 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. (Though not everything is a type. For example, \( \langle e \rangle \) is not a type according to this system even though that is sometimes found in the literature; angle brackets are only introduced for complex types according to this definition.)

We use \( D_\tau \) to signify the set of possible denotations for an expression of type \( \tau \) (for any type \( \tau \)). 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 should be either 1 or 0; \( D_f = \{1, 0\} \). An expression of type \( \langle e, t \rangle \) denotes a function from individuals
to truth values; $D_{(e,t)}$ is the set of functions that take as input an individual, and give a truth value as output. Etc.

The next change has to do with how unary and binary predicates are treated. In $L_1$, we gave the semantics of unary and binary predicates syncategorematically; that is to say, we did not give for example Loves a meaning of its own but rather defined the circumstances under which a formula containing it would be true. In $L_\lambda$, we will treat unary predicates as expressions of type $\langle e, t \rangle$, and binary predicates as expressions of type $\langle e, \langle e, t \rangle \rangle$. They will thus have their own denotations. (In other words, they receive a categorematic treatment.) In fact, technically, we will not have any predicates in this language, of the kind we saw in first-order logic; only functions. For this reason, we will spell them with lowercase letters.

The interpretation of a unary predicate will be a function that takes an individual as input and spits back either 1 ("true") or 0 ("false"). For example, we could assume that the interpretation of the constant bored maps Maggie and Homer to "false", and Bart to "true".

\[
I_1(\text{bored}) = \begin{bmatrix}
\text{Bart} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 0 \\
\text{Maggie} & \rightarrow & 0 
\end{bmatrix}
\]

For the unary predicate happy, let us assume that it denotes the following function:

\[
I_1(\text{happy}) = \begin{bmatrix}
\text{Bart} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 0 \\
\text{Maggie} & \rightarrow & 1 
\end{bmatrix}
\]

**Exercise 1.** Is Bart happy in $M_1$?

(a) What does $I_1(\text{happy})$ give as output when given Bart as an input?
(b) In other words, what is $I_1(\text{happy})(\text{Bart})$?

**Exercise 2.** True or false:

(a) $I_1(\text{bored})(\text{Bart}) = 1$

(b) $I_1(\text{bored})(\text{Bart}) = I_1(\text{happy})(\text{Bart})$

(c) $I_1(n) = \text{Bart}$

(d) $I_1(\text{bored})(I_1(n)) = 1$

(e) $I_1(n) = I_1(j)$

Now for the fun part. A binary predicate will denote a function which, when given an individual, returns another function. This kind of function is the result of Schönfinkelizing the underlying binary relation (also known as Currying; see Heim & Kratzer [1998] 41 for scholarly notes on these terms). For example, given the individual Maggie, the following function returns another function which, given the individual Maggie again, returns 1 (‘true’).

\[
(3) \quad I_1(\text{loves}) = \begin{cases} 
\text{Maggie} & \rightarrow & \begin{cases} 
\text{Maggie} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 0 \\
\text{Bart} & \rightarrow & 0 
\end{cases} \\
\text{Homer} & \rightarrow & \begin{cases} 
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 1 \\
\text{Bart} & \rightarrow & 0 
\end{cases} \\
\text{Bart} & \rightarrow & \begin{cases} 
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 0 \\
\text{Bart} & \rightarrow & 1 
\end{cases} 
\end{cases}
\]
Let us call this function as \( f \). As you can see,

\[
f(Maggie) = \begin{bmatrix}
\text{Maggie} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 0 \\
\text{Bart} & \rightarrow & 0
\end{bmatrix}
\]

Hence \( f(Maggie) \) applied to Maggie is 1. This can be notated as follows (where it is implicit that \( f(Maggie) \) forms a constituent):

\[
f(Maggie)(Maggie) = 1.
\]

But

\[
f(Maggie)(Bart) = 0.
\]

In general, this function corresponds to the ‘same’ relation: It returns true only when both arguments are the same. This describes a situation in which each individual loves him- or herself, and nobody else. Note that this already allows us to give a denotation to verb phrases containing transitive verbs. The meaning of the verb phrase \textit{loves Maggie} can for example be represented as \textit{loves(m)}.

**Exercise 3.**

(a) What is the value of \( I_1(\text{loves}) \) applied to Maggie? Hint: It is a function. Specify the function as a set of input-output pairs (using either set notation or arrow notation).

(b) What is the value of that function (the one you gave in your previous answer) when applied to Maggie?

(c) In other words, what is \( I_1(\text{loves})(Maggie)(Maggie) \)?

(d) So, does Maggie love herself in \( M_1 \)?

Let us suppose that \textit{loves} denotes this ‘same’ function and \textit{kissed} denotes the ‘other’ function, which returns 1 if and only if the two
Typed lambda calculus

arguments are different.

\[ I_1(\text{kissed}) = \begin{bmatrix}
\text{Maggie} & \rightarrow & \begin{bmatrix}
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 1 \\
\text{Bart} & \rightarrow & 1
\end{bmatrix} \\
\text{Homer} & \rightarrow & \begin{bmatrix}
\text{Maggie} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 0 \\
\text{Bart} & \rightarrow & 1
\end{bmatrix} \\
\text{Bart} & \rightarrow & \begin{bmatrix}
\text{Maggie} & \rightarrow & 1 \\
\text{Homer} & \rightarrow & 1 \\
\text{Bart} & \rightarrow & 0
\end{bmatrix}
\] 

Let us assume that the first individual that the function combines with is the person being kissed, and the second individual is the kisser, because a transitive verb combines first with its object and then with its subject. Then this denotation for \textit{kissed} means that Bart and Homer kissed Maggie, Maggie and Bart kissed Homer, and Maggie and Homer kissed Bart. Applied to Maggie, this function yields the function that returns 1 for Bart and Homer:

\[ I_1(\text{kissed})(\text{Maggie}) = \begin{bmatrix}
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 1 \\
\text{Bart} & \rightarrow & 1
\end{bmatrix} \]

And \( I_1(\text{kissed})(\text{Maggie})(\text{Homer}) = 1 \). Now we have a complete specification of all of the values for the basic expressions in the logic.

The introduction of an infinite set of syntactic categories sets the stage for the introduction of a new variable binder called a LAMBDA OPERATOR (or \( \lambda \)-operator), also known as an ABSTRACTION OPERATOR. The lambda operator allows us to describe a wide range of functions without having to name each one. For example:

\[ \lambda x. \text{loves}(x)(m) \]

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

\[ \lambda x. \text{loves}(m)(x) \]
denotes the characteristic function of the set of individuals that love Maggie. You can think of the $\lambda$-operator analogously to predicate notation for building sets. $\lambda x.\text{loves}(x)(m)$ denotes the characteristic function of the set $\{x | \text{Maggie loves } x\}$.

In general, the expression

$$[\lambda x.\phi]$$

denotes a function that returns the value described by $\phi$ when given input $x$. Typically $\phi$ will contain the variable $x$. As it describes the value of the function given an argument, the part that comes after the dot is called the VALUE DESCRIPTION. For example, the value description in the expression

$$\lambda x.\text{loves}(x)(m)$$

is

$$\text{loves}(x)(m).$$

**Exercise 4.** Identify the value description in the following lambda expressions:

1. $\lambda x.\text{happy}(x)$
2. $\lambda x.x$
3. $\lambda y.\lambda x.[\text{loves}(y)(x) \lor \text{loves}(x)(y)]$
4. $\lambda z.\lambda y.\lambda x.\text{introduced}(z)(y)(x)$

If $\phi$ is a formula (type $t$), and $x$ is a variable of type $e$, then $\lambda x.\phi$ will be an expression of type $\langle e, t \rangle$, denoting a function from individuals to truth values. More generally, if $\phi$ is an expression of type $\tau$, and $u$ is a variable of type $\sigma$, then $\lambda u.\phi$ will be an expression of type $\langle \sigma, \tau \rangle$. Such an expression is called a LAMBDA ABSTRACTION.
The functions resulting from abstraction behave just like the functions we are already familiar with. As in \(L_1\), we indicate the arguments of a function using parentheses. This is called application. If \(\pi\) is an expression denoting a function, and \(\alpha\) is an expression that is of the right type to be used as an argument to \(\pi\), then \(\pi(\alpha)\) denotes the result of applying \(\pi\) to \(\alpha\). For example, \(\text{bored}(x)\) denotes the result of applying the function denoted by bored to the value of \(x\). This principle also applies to syntactically complex function-denoting terms formed by lambda abstraction. Thus

\[
[\lambda x. \text{loves}(x)(m)](b)
\]

denotes the result of applying the function ‘is loved by Maggie’ to Bart. This formula is equivalent to the simpler:

\[
\text{loves}(b)(m)
\]

where the \(\lambda\)-binder and the variable have been removed, and we have kept just the value description part, 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 or \(\beta\)-conversion. In general, the result of applying the function to the argument can be described more simply by taking the value description and replacing all occurrences of the lambda-bound variable with the argument. We can formulate the rule more succinctly as follows:

\[
[\lambda x. \phi](\alpha) \text{ can be } \beta\text{-reduced to } \phi[x := \alpha].
\]

where \(\phi[x := \alpha]\) stands for the result of replacing all free occurrences of \(x\) with \(\alpha\) in \(\phi\). Note that \(\beta\)-reduction does not need to be made as an additional stipulation. It's actually a fact that follows from how the semantics of \(\lambda\)-abstraction is defined.

There is one caveat to the rule of \(\beta\)-reduction. It only works if \(\alpha\) does not contain any free variables that occur in \(\phi\). For example,
consider the following expression:

$$\lambda x. \forall y. [\text{loves}(x)(x) \to \text{loves}(y)(x)](y)$$

If we take away $\lambda x$ and substitute $y$ for $x$ then we get:

$$\forall y. [\text{loves}(y)(y) \to \text{loves}(y)(y)]$$

Through overly enthusiastic substitution of $y$ for $x$, the variable $y$ accidentally became bound by the universal quantifier $\forall$. Before we can apply $\beta$-reduction in this case, we must change the name of the variable bound by the universal quantifier thus:

$$[\lambda x. \forall z. [\text{loves}(x)(x) \to \text{loves}(z)(x)]](y)$$

We changed $y$ to $z$ in the scope of the universal quantifier. This is harmless, because a bound variable can be replaced by any other bound variable without a change in meaning. This is the principle of $\alpha$-EQUIVALENCE. Now when we apply $\beta$-reduction, we get the following result:

$$\forall z. [\text{loves}(y)(y) \to \text{loves}(z)(y)]$$

where the variable $y$ no longer clashes with the variable bound by the universal quantifier.

The typed lambda calculus has played an important role in the history of computing, which we will not go into here, as our focus is on natural language semantics. The use of types in that context was crucial to ensure that certain programs terminate, and useful in making sure that the right kinds of operations are combined with the right types of objects (e.g. strings, numbers, etc.). Linguists are typically not so worried about computability, but use types very much in the latter function.

In the version of lambda calculus that linguists have inherited from Montague, variables can range not only over individuals, but also over functions. This, combined with lambda abstraction, allows us to express a wide range of potential meanings 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 `smokes` denotes a function of type `(e, t)`, the characteristic function of the set of smokers, then the meaning of *non-smoker* can be represented as follows:

\[ \lambda x. \neg \text{smokes}(x) \]

This expression denotes a function which takes an individual \(x\) and returns 1 if and only if \(x\) is not a smoker. The meaning of *non-* can be represented as a function that takes as its argument a predicate and then returns a new predicate which holds of an individual if the individual does not satisfy the input predicate. We can represent this function as follows, where \(P\) is a variable that ranges over functions from individuals to truth values:

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

This expression denotes a function which takes as input a predicate \(P\) and returns a new function, denoted by \(\lambda x. \neg P(x)\). The new function takes an individual \(x\) and returns true if and only if \(x\) does not satisfy the predicate. If we apply this function to the predicate `smokes`, then we get the function \(\lambda x. \neg \text{smokes}(x)\). This is the meaning of *non-smoker*, assuming that `smoker` denotes `smokes`.

The \(\lambda\) symbol will also let us give a meaning to phrases like *every cellist* and determiners like *every*. Recall that intuitively, *every* expresses a 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 things that smoke. Let \(X\) and \(Y\) be variables ranging over the characteristic functions of sets (type `(e, t)`). The meaning of *every* can be represented like this:

\[ \lambda X. [\lambda Y. \forall x. [X(x) \to Y(x)]] \]

This expression denotes a function which takes the characteristic function of a set (call it \(X\), and then another (call it \(Y\), and returns 1 (true) if and only if every \(X\) is a \(Y\).
The denotation of every cellist would be the result of applying this function to whatever cellist denotes:

\[ \lambda X. [\lambda Y. \forall x. [X(x) \rightarrow Y(x)]](\text{cellist}) \]

This expression will turn out to be equivalent to:

\[ \lambda Y. \forall x. \text{cellist}(x) \rightarrow Y(x) \]

Thus the meaning of every, applied to the meaning of cellist, is a function that is still hungry for another unary predicate. If we feed it smokes, then we get a sentence that denotes a truth value:

\[ [\lambda Y. \forall x. [\text{cellist}(x) \rightarrow Y(x)]](\text{smokes}) \]

\[ \equiv \forall x. [\text{cellist}(x) \rightarrow \text{smokes}(x)] \]

A crucial point to notice here is that the argument of the function is itself a function. Functions that take other functions as arguments are called higher-order. So what we are dealing with here is no longer first-order logic, but higher-order logic. Pretty fancy.

**Exercise 5.** Download the Lambda Calculator from [http://lambdacalculator.com](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 advertised.

### 4.2 Summary

#### 4.2.1 Syntax of \( L_\lambda \)

Let us now summarize our new logic, \( L_\lambda \), which is a version of the **Typed Lambda Calculus** developed by logician Alonzo Church. The **Types** are defined recursively as follows: \( e \) and \( t \) are both
Typed lambda calculus

If $\sigma$ and $\tau$ are types, then $\langle \sigma, \tau \rangle$ is a type; nothing else is a type. A formula is an expression of type $t$.

The set of expressions of type $\tau$, for any type $\tau$, is defined recursively as follows:

1. **Basic Expressions** *(now more general!)*
   For each type $\tau$,
   
   (a) the **non-logical constants** of type $\tau$ are they symbols of the form $c_{n,\tau}$ for each natural number $n$.
   
   (b) the **variables** of type $\tau$ are the symbols of the form $\nu_{n,\tau}$ for each natural number $n$.

2. **Predication** *(cf. ‘Atomic Formulas’ – more general!)*
   For any types $\sigma$ and $\tau$, if $\alpha$ is an expression of type $\langle \sigma, \tau \rangle$ and $\beta$ is an expression of type $\tau$ then $\alpha(\beta)$ is an expression of type $\tau$.

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

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

5. **Binary Connectives**
   If $\phi$ and $\psi$ are formulas, then so are $\neg \phi$, $\phi \land \psi$, $\phi \lor \psi$, $\phi \rightarrow \psi$, and $\phi \leftrightarrow \psi$.

6. **Quantification**
   If $\phi$ 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 $\alpha$ is an expression of type $\tau$ and $u$ is a variable of type $\sigma$ 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 (e.g. $\forall x$ or $\exists x$) extends as far to the right as
possible. So, for example, \( \lambda x. [P(x) \land Q(x)] \) can be rewritten as \( \lambda x. P(x) \land Q(x) \). However, we will typically retain brackets in these cases. Recall also that we drop the dot whenever we have multiple binders in a row, so we can write, e.g. \( \lambda x\lambda y. \text{Admires}(x, y) \).

To further reduce clutter, we will add the following abbrevatory convention, so we can for example write \( \pi(\lambda x. x + 1) \) rather than \( \pi([\lambda x. x + 1]) \): Square brackets that are immediately embedded inside parentheses can be dropped.

Note that

\[
\text{Loves}(h, m)
\]

is not a formula anymore according to our rules. The ‘Atomic Formulas’ rule has been replaced by the rule of Predication, so this idea would be expressed as

\[
\text{loves}(m)(h)
\]

instead. But sometimes it feels easier to use the style that is familiar from first-order logic, and one could in principle augment the language to include formulas like Loves\((h, m)\). The Lambda Calculator does this, treating Loves as a constant of type \( (e \times e, t) \), where \( e \times e \) is the type of ordered pairs of individuals. So \( \pi(\alpha, \beta) \) is well-formed when \( \pi \) designates a function that expects an ordered pair as input (a so-called PRODUCT TYPE). To incorporate this, we must expand the range of types so that of \( \sigma \) is a type and \( \tau \) is a type, then \( \sigma \times \tau \) is a type.

**Exercise 6.** 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 \( v_{0,(e,t)} \), \( Q \) is \( v_{1,(e,t)} \), and \( X \) is \( v_{2,(e,t)} \)
Typed lambda calculus

- $R$ is $v_0, e \times e, t$
- $\nu$ is $c_{0,e}$ and $d$ is $c_{1,e}$

1. $[\lambda x. P(x)](a)$
2. $[\lambda x. P(x)](a)$
3. $[\lambda x. R(y, a)]$
4. $[\lambda x. R(y, a)](b)$
5. $[\lambda x. R(x, a)](b)$
6. $[\lambda x y. R(x, y)](b)$
7. $[\lambda x y. R(x, y)](b)(a)$
8. $[\lambda x. [\lambda y. R(x, y)](b)](a)$
9. $[\lambda X. \exists x. [P(x) \land X(x)]](\lambda y. R(a, y))$
10. $[\lambda X. \exists x. [P(x) \land X(x)]](\lambda x. R(a, x))$
11. $[\lambda X. \exists x. [P(x) \land X(x)]](\lambda y. R(y, x))$
12. $[\lambda X. \exists x. [P(x) \land X(x)]](Q)$
13. $[\lambda X. \exists x. [P(x) \land X(x)]](X)$
14. $[\lambda X. \exists x. [P(x) \land X(x)]](\lambda x. Q(x))$
15. $[\lambda y \lambda x. R(y, x)](a)$

For each of the above, answer the following questions:

(a) Is it a well-formed expression of $L_\lambda$ (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 ($\lambda$-converted) expression which is equivalent to it. Use $\alpha$-equivalence (relettering of bound variables) if necessary to avoid variable clash.

You can check your answers using the Lambda Calculator.

**Exercise 7.** Identify the type of each of the following. Assume that man and mortal are constants of type $\langle e, t \rangle$. Check your work using the Lambda Calculator.

1. $\lambda y. y$
2. $\lambda x. P(x)$
3. $P$
4. $a$
5. $x$
6. $P(x)$
7. $[\lambda x. P(x)](a)$
8. $P(a)$
9. $R(x, y)$
10. $\lambda x. R(x, a)$
11. $\lambda y \lambda x. R(y, x)$
12. $[\lambda y \lambda x. R(y, x)](a)$
13. $[\lambda x. R(y, a)](b)$
14. $R(a, b)$
15. $\lambda x. [P(x) \land Q(x)]$
16. $[\lambda x. P(x) \land Q(x)](a)$
17. $\lambda x \lambda y. [R(y)(a) \land Q(x)]$
18. $\lambda P. P$
19. $\lambda P. P(a)$
20. $\exists x. P(x)$
21. $\lambda P. \exists x. P(x)$
22. $[\lambda P. \exists x. P(x)](\text{man})$
23. $\exists x. \text{man}(x)$
24. $\lambda P. \forall x. P(x)$
25. $[\lambda P. \forall x. P(x)](\text{mortal})$
26. $\neg \text{mortal}(x)$
27. $\lambda x. \neg \text{mortal}(x)$
28. $\lambda P \lambda x. \neg P(x)$
29. $[\lambda P \lambda x. \neg P(x)](\text{mortal})$
30. $\lambda x. \neg \text{mortal}(x)$
31. $[\lambda x. \neg \text{mortal}(x)](a)$
32. $\neg \text{mortal}(a)$
33. $\lambda Q. \forall x. [\text{man}(x) \rightarrow Q(x)]$
Exercise 8. 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. $[\lambda x.x](a)$
2. $[\lambda P.P](\text{man})$
3. $[\lambda P.\exists x.P(x)](\text{man})$
4. $[\lambda x.P(x)](a)$
5. $[\lambda x.P(x)]$
6. $[\lambda y\lambda x.R(y,x)](a)$
7. $[\lambda x.R(y,a)](b)$
8. $[\lambda x.[P(x) \land Q(x)]](a)$
9. $[\lambda P.\exists x.P(x)](\text{man})$
10. $[\lambda P. \forall x. P(x)](\text{mortal})$

11. $\lambda x. \neg \text{mortal}(x)$

12. $[\lambda P \lambda x. \neg P(x)](\text{mortal})$

13. $[\lambda x. \neg \text{mortal}(x)](a)$

14. $[\lambda Q. \forall x.[\text{man}(x) \rightarrow Q(x)]](\text{mortal})$

15. $[\lambda P \lambda Q. \forall x. [P(x) \rightarrow Q(x)]](\text{man})$

16. $[\lambda P \lambda Q. \forall x[P(x) \rightarrow Q(x)]](\text{man})(\text{mortal})$

17. $[\lambda x. P(x) \land Q(x)](a)$

18. $[\lambda x \lambda y.[R(y, a) \land Q(x)]](a)(b)$

19. $[\lambda x. \exists y. R(x, y)](y)$

20. $[\lambda x. a](b)$

21. $[\lambda x. [P(x) \rightarrow \exists x. R(b, x)]](a)$

22. $[\lambda Q. \forall x[\text{mortal}(x) \rightarrow Q(x)]](\lambda x[\text{mortal}(x)])$

23. $[\lambda Q. \exists P. \forall x[P(x) \rightarrow Q(x)]](\text{mortal})$

24. $[\lambda P \lambda x. \neg P(x)](\lambda x[\text{mortal}(x)])$

25. $[\lambda P \lambda x. P(x)](\lambda x[\neg \text{mortal}(x)])$

You can check your answers using the Lambda Calculator.

4.2.2 Semantics

As in $L_1$, the semantic values of expressions in $L_\Lambda$ depend on a model and an assignment function. As in $L_1$, a model $M = (D, I)$
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 \( a \), \( I \) assigns an object of type \( a \) to every non-logical constant of type \( a \).

Types are associated with domains. The domain of individuals \( D \) 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 \( b \), \( D\{a,b\} \) is the domain of functions from \( D_a \) to \( D_b \).

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 \( \alpha \) is a non-logical constant, then \( \llbracket \alpha \rrbracket^{M,g} = I(\alpha) \).
   
   (b) If \( \alpha \) is a variable, then \( \llbracket \alpha \rrbracket^{M,g} = g(\alpha) \).

2. **Predication**
   
   If \( \alpha \) is an expression of type \( \langle a, b \rangle \), and \( \beta \) is an expression of type \( a \), then \( \llbracket \alpha(\beta) \rrbracket = \llbracket \alpha \rrbracket(\llbracket \beta \rrbracket) \).

3. **Equality**
   
   If \( \alpha \) and \( \beta \) are terms, then \( \llbracket \alpha = \beta \rrbracket^{M,g} = 1 \) iff \( \llbracket \alpha \rrbracket^{M,g} = \llbracket \beta \rrbracket^{M,g} \).

4. **Negation**
   
   If \( \phi \) is a formula, then \( \llbracket \neg \phi \rrbracket^{M,g} = 1 \) iff \( \llbracket \phi \rrbracket^{M,g} = 0 \).

5. **Binary Connectives**
   
   If \( \phi \) and \( \psi \) are formulas, then:

   (a) \( \llbracket \phi \land \psi \rrbracket^{M,g} = 1 \) iff \( \llbracket \phi \rrbracket^{M,g} = 1 \) and \( \llbracket \psi \rrbracket^{M,g} = 1 \).

   (b) \( \llbracket \phi \lor \psi \rrbracket^{M,g} = 1 \) iff \( \llbracket \phi \rrbracket^{M,g} = 1 \) or \( \llbracket \psi \rrbracket^{M,g} = 1 \).

   (c) \( \llbracket \phi \rightarrow \psi \rrbracket^{M,g} = 1 \) iff \( \llbracket \phi \rrbracket^{M,g} = 0 \) and \( \llbracket \psi \rrbracket^{M,g} = 1 \).

   (d) \( \llbracket \phi \leftrightarrow \psi \rrbracket^{M,g} = 1 \) iff \( \llbracket \phi \rrbracket^{M,g} = \llbracket \psi \rrbracket^{M,g} \).
6. **Quantification**

(a) If $\phi$ is a formula and $v$ is a variable of type $a$ then $\lbrack \forall v. \phi \rbrack^{M,g} = 1$ iff for all $k \in D_a$:

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

(b) If $\phi$ is a formula and $v$ is a variable of type $a$ then $\lbrack \exists v. \phi \rbrack^{M,g} = 1$ iff there is an individual $k \in D_a$ such that that:

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

7. **Lambda Abstraction**

If $\alpha$ is an expression of type $a$ and $u$ a variable of type $b$ then $\lbrack \lambda u. \alpha \rbrack^{M,g}$ is that function $h$ from $D_b$ into $D_a$ such that for all objects $k$ in $D_b$, $h(k) = \lbrack \alpha \rbrack^{M,g[u \mapsto k]}$.

**Exercise 9.**

(a) Partially define a model for $L_\lambda$ giving values (denotations) to the constants loves, n, and d.

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

**Exercise 10.** 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, where $\text{aunt}(x, y)$ holds between $x$ and $y$ if and only if $x$ is $y$’s aunt. Thus a sentence like *Sue is Alex’s aunt* could be represented as $\text{aunt}(\text{sue}, \text{alex})$. But consider a sentence like *Sue is an*
Typed lambda calculus

*aunt now!* Such a sentence might be taken to express an existential claim like $\exists x. \text{aunt}(\text{sue}, x)$. 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: $\lambda y. \exists x. \text{aunt}(\text{sue}, x)$. 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 $L_\lambda$, 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 11.** 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 $L_\lambda$. 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 12.** 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 \( L_\lambda \) of type \( \langle \langle e, \langle e, t \rangle \rangle, \langle e, t \rangle \rangle \) which produces this sort of meaning from a two-place predicate. (Adapted from Dowty et al. (1981), Problem 4-7, p. 97.)

**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.
Translating to lambda calculus

Now we will begin to talk about natural language expressions and their denotations. To do so, we will translate expressions of English into expressions of $L_\lambda$, with statements like this:

\[ \text{snores} \sim \lambda x. \text{snores}(x) \]

where $\sim$ signifies the 'translates to' relation. If we can assume that there is a designated, unique $L_\lambda$ expression to which a given natural language expression $\alpha$ is translated, then we can write $\langle \alpha \rangle$ to denote 'the (primary) translation of $\alpha$'.

(1) **Notation convention**

$\langle \alpha \rangle$ denotes the primary translation of $\alpha$ into $L_\lambda$.

For example:

\[ \langle \text{snores} \rangle = \lambda x. \text{snores}(x) \]

Any expression in $L_\lambda$ that is semantically equivalent to $\langle \alpha \rangle$ but syntactically distinct is technically not appropriately called $\langle \alpha \rangle$, because $\langle \alpha \rangle$ refers to a particular, syntactically specified expression\(^1\). We will often work with simplified ($\beta$-reduced) meaning representations, which are equivalent to, but not identical to, the primary translations\(^2\).

---

\(^1\)So-called ‘type-shifting rules’ have been treated in indirect interpretation frameworks as providing additional translations; see for example Partee & Rooth (1983a). Here, we will represent type-shifting operations with extra syntactic node, as is the typical practice in the Lambda Calculator.

\(^2\)It is very common in formal semantics to write things like:
We then let the semantics of the English expressions be inherited from the semantics of lambda calculus. 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.

This is like what Montague (1974b) did in his famous work entitled *The Proper Treatment of Quantification in Ordinary English* (‘PTQ’ for short). There, unlike in *English as a Formal Language* (Montague, 1974a), he specified a set of principles for translating English into his own version of lambda calculus called Intensional Logic. He was very clear that this procedure was only meant to be a convenience; one could in principle specify the meanings 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 lambda calculus. Nevertheless, the expressions of lambda calculus are not themselves the denotations. Rather, we have two object languages, English (a natural language) and lambda calculus (a formal language), and we are translating from the natural language to the formal language, and specifying the semantics of the formal language in our metalanguage (which is also English, mixed with talk of sets and relations).³

\[
\text{[snores]} = \lambda x. \text{snores}(x)
\]

It is not entirely straightforward how to interpret expressions like this. The use of denotation brackets around the natural language expression clearly suggests that a kind of direct interpretation is intended. The intention appears to be to use the expression on the right (as opposed to mentioning it, as we do in indirect translation) to describe the semantic value of the expression, in other words to name a particular function. But even if we assume that the expression is to be interpreted ‘in the standard way’, it cannot be interpreted in the absence of a given model (and assignment function, if there are free variables), and it is not obvious how these come into play. Moreover, using indirect translation gives us better control over the representation language; this way, we can define new logical symbols, change how the logical symbols are interpreted, and add new evaluation parameters.

³An important difference between the tact we are taking here and the one
In this section, we will explore the limits of Functional Application, and define a larger fragment. To do so, we will need a few syntax rules.

(2) **Syntax**

\[
\begin{align*}
S & \rightarrow \text{DP VP} \\
S & \rightarrow \text{Neg S} \\
S & \rightarrow \text{S JP} \\
J & \rightarrow J S \\
VP & \rightarrow V \text{ (DP|AP|PP)} \\
AP & \rightarrow A \text{ (PP)} \\
DP & \rightarrow D \text{ (NP)} \\
NP & \rightarrow N \text{ (PP)} \\
NP & \rightarrow A \text{ NP} \\
PP & \rightarrow P \text{ DP}
\end{align*}
\]

The vertical bar | separates alternative possibilities, and the parentheses signify optionality, so the VP rule means that a VP can consist solely of a verb, or of a verb followed by an NP, or of a verb followed by an AP, etc.

The terminal nodes of the syntax trees produced by these syntax rules may be populated by the following words:

(3) **Lexicon**

\[\text{J}: \text{and, or}\]

\[\text{taken in Heim & Kratzer's 1998 textbook is that here the} \lambda \text{ symbol is used as an expression of lambda calculus (its original use), whereas in Heim and Kratzer the} \lambda \text{ symbol is part of the meta-language, as an abbreviation for describing functions. One should carefully distinguish between these two ways of using it.}\]

\[\text{Here we are adopting the 'DP-hypothesis', where DP stands for 'Determiner Phrase', and we assume that the head of a phrase like the car is the determiner the. Since phrases like the car and proper names like Homer are of the same category, this means that Homer is also a DP. We assume therefore that proper names are of category D, like articles. The difference is that proper names are 'intransitive', so they don't combine with an NP.}\]
Translating to lambda calculus

Neg: *it is not the case that*

V: *smokes, loves, is*

A: *lazy, proud*

N: *drunkard, baby, kid, zebra*

D: *the, a, every, some, no*

D: *Bart, Maggie, Homer, everybody, somebody, nobody*

P: *of, with*

**Exercise 1.** Which of the following strings are sentences of the fragment of English that we have defined (modulo sentence-initial capitalization)?

(a) Homer Simpson loves everybody.

(b) Some drunkard smokes every lazy zebra.

(c) Maggie is not a zebra.

(d) Homer Simpson is.

(e) Homer is.

(f) Homer is a baby zebra.

(g) Somebody is proud of the kid.

(h) A zebra loves proud of Bart.

(i) The proud zebra of Homer loves every lazy lazy lazy lazy drunkard.

(j) Bart smokes with nobody.

(k) Maggie and Bart are with Homer.

(l) Maggie is with Homer and Bart is with Homer.
Note: In the trees below, sometimes we “prune” non-branching nodes. For example, we might write:

\[
\begin{array}{c}
\text{DP} \\
\mid \\
\text{Homer}
\end{array}
\]

instead of

\[
\begin{array}{c}
\text{DP} \\
\mid \\
\text{D} \\
\mid \\
\text{Homer}
\end{array}
\]

Now that we have defined the syntax of our fragment of English, we need to specify how the expressions generated by these syntax rules are interpreted. To do so, we will translate them into expressions of \( L_\lambda \). 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. We will build up a theory of how these translations work by considering a series of examples.

### 5.1 Fun with Functional Application

#### 5.1.1 Homer loves Maggie

Let us first consider how to analyze a simple transitive sentence like \textit{Homer loves Maggie}. We will represent the meaning of the verb \textit{loves} as follows:

\[
(4) \quad \langle \text{loves} \rangle = \lambda y \lambda x. \text{loves}(y)(x)
\]

The expression \( \langle \text{loves} \rangle \) can be read “the translation of the word \textit{loves} into \( L_\lambda \)”. We refer to this mapping from a word to a meaning representation in the formal language as a \textsc{lexical entry} for the verb \textit{loves}. Note that according to the way we have set things
up, we could simply have written loves on the right-hand side of the equals sign instead of $\lambda y \lambda x.\text{loves}(y)(x)$. But the longer style brings out the fact that we are dealing with a function of type $\langle e, \langle e, t \rangle \rangle$ and is more in line with existing conventions in semantics, following [Heim & Kratzer (1998)], where every argument corresponds to a $\lambda$-abstraction. Let us assume further that Homer is translated as $h$ and that Maggie is translated as $m$.

- $\langle Homer \rangle = h$
- $\langle Maggie \rangle = m$

To put together the meanings of complex expressions, we depend primarily on the rule of Functional Application, which simply takes a function and applies it to an appropriate argument:

**Composition Rule 1. Functional Application** (FA)

Let $\gamma$ be a tree whose only two subtrees are $\alpha$ and $\beta$. If $\langle \alpha \rangle$ is of type $\langle \sigma, \tau \rangle$ and $\langle \beta \rangle$ is of type $\sigma$, then:

$$\langle \gamma \rangle = \langle \alpha \rangle(\langle \beta \rangle)$$

For example, the tree $[VP [V loves] [NP Maggie]]$ has two subtrees, $[V loves]$ and $[NP Maggie]$. Since the translation of first one denotes a function that applies to the other, we can use FA to put together the meanings of these two expressions. So

$$\langle [VP [V loves] [NP Maggie]] \rangle = \langle [V loves] \rangle(\langle [NP Maggie] \rangle)$$

The next step is to give translations to $[V loves]$ and $[NP Maggie]$. For this, we will also need a rule for non-branching nodes. (We have specified the translation for loves but not for $[V loves]$.)
Composition Rule 2. Non-branching Nodes (NN)
If $\beta$ is a tree whose only daughter is $\alpha$, then $\langle \beta \rangle = \langle \alpha \rangle$.

*Homer loves Maggie* can then be analyzed as follows.

(5)

\[
\begin{array}{c}
S \\
\text{loves}(m)(h) \\
\text{DP} \\
\lambda x.\text{loves}(m)(x) \\
\text{VP} \\
\langle e, t \rangle \\
\text{DP} \\
\langle e, \langle e, t \rangle \rangle \\
\lambda y \lambda x.\text{loves}(y)(x) \\
\text{V} \\
\text{loves} \\
\text{Maggie}
\end{array}
\]

Via Functional Application, the transitive verb *loves* combines with the object *Maggie*, and the VP combines with the subject *Homer*. 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 *Homer* (an individual) to produce a truth value. Note that we have not shown the primary translations at every node, but rather have applied $\beta$-reduction in order to simplify the logical expressions wherever possible.

**Exercise 2.** 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.
5.1.2  *Homer is lazy*

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

![Syntactic Structure Diagram](image)

We will continue to assume that the proper name *Homer* is translated as the constant $h$, of type $e$. We can assume that *lazy* denotes a function of type $(e, t)$, the characteristic function of a set of individuals (those that are lazy). Let us use *lazy* as an abbreviation for a constant of type $(e, t)$, and translate *lazy* thus.
(6) \[ \langle \text{lazy} \rangle = \lambda x . \text{lazy}(x) \]

Technically, again, according to the way we have set things up, we could write the same thing more simply by simply writing lazy. But the longer style brings out the fact that we are dealing with a function with one argument.

What is the meaning of *is*? Besides signaling present tense, it does not seem to accomplish more than to link the predicate ‘lazy’ with the subject of the sentence. Since we have not started dealing with tense yet, we will ignore the former function and focus on the latter. We 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 \( <e, t> \), and returns that same function.

(7) \[ \langle \text{is} \rangle = \lambda P . P \]

This implies that *is* denotes a function that takes as its first argument another function \( P \), where \( P \) is of type \( <e, t> \), and returns \( P \).

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

(8) \[
\begin{array}{c}
S \\
  \downarrow \\
  t \\
  \downarrow \\
  \text{lazy}(h) \\
  \uparrow \\
  \text{DP} \\
  \downarrow \\
  e \\
  \downarrow \\
  h \\
  \uparrow \\
  \text{VP} \\
  \downarrow \\
  \langle e, t \rangle \\
  \downarrow \\
  \lambda x . \text{lazy}(x) \\
  \uparrow \\
  \text{Homer} \\
  \downarrow \\
  \langle \langle e, t \rangle, \langle e, t \rangle \rangle \\
  \downarrow \\
  \lambda P . P \\
  \downarrow \\
  \text{is} \\
  \downarrow \\
  \lambda x . \text{lazy}(x) \\
  \downarrow \\
  \text{lazy}
\end{array}
\]
Each node shows the syntactic category, the semantic type, and a fully $\beta$-reduced translation to lambda calculus. In this case, Functional 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 ($Homer$, $is$, and lazy).

**Exercise 3.** Assume the syntax rules have been changed so as to allow this structure. Then provide an analysis of the following sentence that captures the fact that it contradicts $Homer$ is lazy, giving the semantic type and a fully $\beta$-reduced translation at each node.

```
S
  DP
    Homer
  VP
    V
      is
    NegP
      Neg
        not
      AP
        A
          lazy
```

### 5.1.3 Homer is with Maggie

Like adjectives, prepositional phrases can also serve as predicates, as in, for example, *Homer is with Maggie*. Let us introduce the constant with, a function of type $\langle e, \langle e, t \rangle \rangle$, and translate *with* as follows:

$$
\langle with \rangle = \lambda y \lambda x. with(y)(x)
$$

Here is an overview of how the derivation will go. Via Functional Application, the preposition *with* combines with its object *Maggie*, and the resulting PP combines with *is* to form a VP which
combines with the subject *Homer*. 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 *Homer* to produce a truth value.

Exercise 4. Derive the translation into \( L_\lambda \) for *Homer is with Maggie* by giving a fully \( \beta \)-reduced translation for each node.

Exercise 5. Suppose model \( M_1 \) is defined such that the following holds:

\[
[\text{with}]^{M_1} = \begin{cases} 
\text{Bart} & \rightarrow & \begin{cases} 
\text{Bart} & \rightarrow & 0 \\
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 1 
\end{cases} \\
\text{Maggie} & \rightarrow & \begin{cases} 
\text{Bart} & \rightarrow & 0 \\
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 0 
\end{cases} \\
\text{Homer} & \rightarrow & \begin{cases} 
\text{Bart} & \rightarrow & 1 \\
\text{Maggie} & \rightarrow & 0 \\
\text{Homer} & \rightarrow & 0 
\end{cases}
\end{cases}
\]
and suppose that $[d]^{M_1} = \text{Homer}$ and $[m]^{M_1} = \text{Maggie}$. What truth value does the translation for \textit{Homer is with Maggie} have in $M_1$? Explain your answer.

### 5.1.4 \textit{Homer is proud of Maggie}

Like prepositions, adjectives can denote functions of type $\langle e, \langle e, t \rangle \rangle$. \textit{Proud} is an example; in \textit{Homer is proud of Maggie}, the adjective \textit{proud} expresses a relation that holds between Homer and Maggie. We can capture this by assuming that \textit{proud} translates to the constant \textit{proud} of type $\langle e, \langle e, t \rangle \rangle$, denoting a function that takes two arguments, first a potential object of pride (such as Maggie), then a potential bearer of such pride (e.g. Homer), and then returns if the pride relation holds between them.

In contrast to \textit{with}, the preposition \textit{of} does not seem to signal a two-place relation in this context. We therefore assume that \textit{of} is a function word like \textit{is}, and also denotes an identity function. Unlike \textit{is}, however, we will treat \textit{of} as an identity function that takes an individual and returns an individual, so it will be of category $\langle e, e \rangle$.

\begin{equation}
\llbracket \text{of} \rrbracket = \lambda x. x
\end{equation}

So the adjective phrase \textit{proud of Maggie} 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 *Homer is proud of Maggie*.

Exercise 7. What would go wrong if we were to assume that *of* was of type $\langle e, \langle e, t \rangle \rangle$, like *with*?

5.1.5 *Homer is a drunkard*

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

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.

(11) \[ \langle a \rangle = \lambda P . P \]

With these assumptions, the derivation will go as follows.
Exercise 8. Give fully \( \beta \)-reduced translations at each node of the tree for *Homer is a drunkard*.

Exercise 9. Can we treat \( a \) as \( \langle (e, t), (e, t) \rangle \) in a sentence like *A drunkard loves Maggie*? Why or why not?

Exercise 10. Assume that *radical* and *communist* are both of type \( \langle e, t \rangle \), following the style we have developed so far. Is it possible to assign truth conditions to the following sentence using those assumptions? Why or why not?
Exercise 11. 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 analyze the following tree. 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 12.** 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 \textit{self1}, \textit{self2}, and \textit{self3}. Assuming the syntactic structure given, give a denotation for each of these morphemes.

(a) For the sentence \textit{Carlos self1-shaves}, make the structure below yield the meaning ‘Carlos shaves himself’ by supplying the denotation of \textit{self1}.

(b) For the sentence \textit{Carlos self2-introduced Paco}, make the structure below yield the meaning ‘Carlos introduced Paco to Carlos (himself)’ by supplying the denotation of \textit{self2}.
(c) For the sentence *Carlos self3-introduced Paco*, make the structure below yield the meaning ‘Carlos introduced Paco to Paco (himself)’ by supplying the denotation of *self3*.

![Tree diagram](image)

Make sure that your denotations work not just for sentences involving Carlos and Paco, but arbitrary proper names. (Exercise due to Maribel Romero.)

5.1.6 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

\[
\begin{align*}
S & \rightarrow \text{DP VP} \\
S & \rightarrow \text{Neg S} \\
\text{VP} & \rightarrow \text{V (DP|AP|PP)} \\
\text{AP} & \rightarrow \text{A (PP)} \\
\text{DP} & \rightarrow \text{D (NP)} \\
\text{NP} & \rightarrow \text{N (PP)} \\
\text{NP} & \rightarrow \text{A NP} \\
\text{PP} & \rightarrow \text{P DP}
\end{align*}
\]
Lexicon

Neg: it is not the case that
V: smokes, loves, is
A: lazy, proud
N: drunkard, baby, kid, zebra
D: the, a, every, some, no
D: Bart, Maggie, Homer, everybody, somebody, nobody
P: of, with

Composition Rules

• **Functional Application** (FA)
  Let \( \gamma \) be a tree whose only two subtrees are \( \alpha \) and \( \beta \). If \( \langle \alpha \rangle \) is of type \( \langle \sigma, \tau \rangle \) and \( \langle \beta \rangle \) is of type \( \sigma \), then:

  \[ \langle \gamma \rangle = \langle \alpha \rangle (\langle \beta \rangle) \]

• **Non-branching Nodes** (NN)
  If \( \beta \) is a tree whose only daughter is \( \alpha \), then \( \langle \beta \rangle = \langle \alpha \rangle \).

Lexical entries

• \( \langle Homer \rangle = d \)
• \( \langle Maggie \rangle = m \)
• \( \langle loves \rangle = \lambda y \lambda x. \text{loves}(y)(x) \)
• \( \langle lazy \rangle = \lambda x. \text{lazy}(x) \)
• \( \langle is \rangle = \lambda P. P \)
• \( \langle of \rangle = \lambda x. x \)
• \( \langle a \rangle = \lambda P. P \)
• \( \langle with \rangle = \lambda y \lambda x. \text{with}(y)(x) \)
Exercise 13. Extend this fragment to assign representations in $L_\lambda$ to the following sentences. You may have to modify or add to the syntax rules, the lexicon, the composition rules, or the lexical entries. (Although you are free to add to the composition rules, the exercises can be solved without doing so.) For each sentence, give a parse tree with a fully beta-reduced representation at each node.

1. Homer is a fan of Maggie.
2. Homer is Maggie’s fan.
3. It is not the case that Maggie smokes.
4. Homer smokes and Maggie drinks.
5. Homer smokes and drinks.

5.2 Predicate Modification

5.2.1 *Homer is a lazy drunkard*

Now let us consider the sentence *Homer is a lazy drunkard*. This one presents a bit of a problem. According to our lexical entry for lazy above, lazy denotes a function of type $\langle e, t \rangle$, and so does drunkard. These two expressions are sisters in the tree, but neither one denotes a function that has the denotation of the other in its domain. So we can’t use Functional Application to combine them, and so far, we have no other rules for combining expressions together.
If we want to use Functional Application here, we need `lazy` to be a function of type `⟨⟨e, t⟩, ⟨e, t⟩⟩`. We can do this using a \textit{type-shifting rule} which introduces a possible translation of type `⟨⟨e, t⟩, ⟨e, t⟩⟩` for every translation of type `⟨e, t⟩` (Partee, 1995, p. 29).

There are two technical ways we could implement type-shifting. One would be to specify that, in addition to the primary translation for `lazy` of type `⟨e, t⟩`, there is in addition a secondary translation of type `⟨⟨e, t⟩, ⟨e, t⟩⟩`. Another way to do it would be to assume that type-shifters are present in the syntax tree. So the type-shifted version of `lazy` will technically be like this syntactically:
(14) A
\[\{ (e, t), (e, t) \} \]
MOD A
\[\{ e, t \} \]
lazy

Under this second style, we don't have to bother with the distinction between primary and secondary translations, and it is consistent with how type-shifting is typically handled in the Lambda Calculator. This is the style we will follow here. However, we will use a special notation for trees with type-shifters in order to capture the intuition that the type-shifting operation induces a transformation of the meaning:

(15) A
\[\{ (e, t), (e, t) \} \]
\[\uparrow_{\text{MOD}}\]
A
\[\{ e, t \} \]
lazy

The type-shifting rule that we use is defined as follows:

**Type-Shifting Rule 1.** Predicate-to-modifier shift (MOD)
If \(\llangle \alpha \rrangle\) is of category \(\{ e, t \}\), then:

\[
\llangle \text{MOD } \alpha \rrangle = \lambda P \lambda x. [\llangle \alpha \rrangle (x) \land P(x)]
\]
as well (as long as \(P\) and \(x\) are not free in \(\llangle \alpha \rrangle\); in that case, use different variables of the same type).

By this rule, we can derive that:

\[
\llangle \text{MOD } \text{lazy} \rrangle = \lambda P \lambda x. [\text{lazy}(x) \land P(x)]
\]
so *Homer is a lazy drunkard* gets the following analysis:

![Diagram of the analysis of "Homer is a lazy drunkard"]

This is essentially Montague’s strategy for dealing with adjectives that modify nouns. In fact, *all* adjectives are \((e, t)\) for him. For a sentence like *Homer is lazy*, Montague assumes that the adjective combines with a silent noun. We would have to augment our syntax rules to allow for this, but if we did, then it would become mysterious why *Homer is lazy drunkard* is not grammatical in English, and why we can’t have silent nouns all over the place. So it seems necessary to assume that adjectives can at least sometimes be of type \((e, t)\).
**Exercise 14.** There is another possible type-shifting analysis, where the version with the more complicated type \((\langle e, t \rangle, \langle e, t \rangle)\) is basic, and the predicative, \(\langle e, t \rangle\) version is derived through a type-shifting rule. Specify an appropriate such rule.\[5\]

Instead of using a type-shifting rule to interpret attributive adjectives, another strategy is to let go of Frege's conjecture (that Functional Application is basically the only composition rule), and accept another composition rule. This rule would take two predicates of type \(\langle e, t \rangle\), and combine them into a new predicate of type \(\langle e, t \rangle\). The new predicate would hold of anything that satisfied both of the old predicates. This is how Predicate Modification is defined:

**Composition Rule 3. Predicate Modification** (PM)
If \(\langle \alpha \rangle\) and \(\langle \beta \rangle\) are of type \(\langle e, t \rangle\), and \(\gamma\) is a tree whose only two subtrees are \(\alpha\) and \(\beta\), then:

\[
\langle \gamma \rangle = \lambda u. [\langle \alpha \rangle(u) \land \langle \beta \rangle(u)]
\]

where \(u\) is a variable of type \(e\) that does not occur free in \(\langle \alpha \rangle\) or \(\langle \beta \rangle\).

This gives us the following derivation for the NP:

\[
\text{(17) NP} \quad \lambda x. [\text{lazy}(x) \land \text{drunkard}(x)]
\]

\[
\begin{array}{c}
\text{NP} \\
\langle e, t \rangle \\
\lambda x. [\text{lazy}(x) \land \text{drunkard}(x)]
\end{array}
\]

\[
\begin{array}{c}
A \\
\langle e, t \rangle \\
\text{lazy}
\end{array} \quad \begin{array}{c}
\text{NP} \\
\langle e, t \rangle \\
\text{drunkard}
\end{array}
\]

\[
\begin{array}{c}
lazy \\
\text{drunkard}
\end{array}
\]
Exercise 15. Consider the sentence *Maggie is a lazy baby.* 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 β-reduced translation. (Feel free to use the Lambda Calculator for this.)

Exercise 16. Identify the types of the following expressions:

(a) \( \lambda x \lambda y . \text{in}(y, x) \)
(b) \( \lambda x . x \)
(c) \( \lambda x . \text{city}(x) \)
(d) \( \text{texas} \)
(e) \( \lambda y . \text{in}(y, \text{texas}) \)
(f) \( \lambda f . f \)
(g) \( \lambda x \lambda y . \text{fond-of}(y, x) \)

Assume:

- \( x \) and \( y \) are variables of type \( e \), and \( f \) is a variable of type \( (e, t) \).
- Any constant that appears with an argument list of length 1 (e.g. \( \text{city} \)) is a unary predicate, and any constant that appears with an argument list of length 2 (e.g. \( \text{in} \)).
- Any constant that appears without an argument list (e.g. \( \text{texas} \)) is type \( e \).
The following exercises are adapted from Heim & Kratzer (1998), via Lucas Champollion's adaptation of them for the Lambda Calculator.

Exercise 17. In addition to the ones given above, adopt the following lexical entries, using the same assumptions about types as in the previous exercise:

1. \( \langle \text{cat} \rangle = \lambda x. \text{cat}(x) \)
2. \( \langle \text{city} \rangle = \lambda x. \text{city}(x) \)
3. \( \langle \text{gray} \rangle = \lambda x. \text{gray}(x) \)
4. \( \langle \text{gray}_2 \rangle = \lambda f \lambda x. \text{gray}(x) \cap f(x) \)
5. \( \langle \text{in} \rangle = \lambda x \lambda y. \text{in}(y, x) \)
6. \( \langle \text{in}_2 \rangle = \lambda y \lambda f \lambda x. f(x) \cap \text{in}(y)(x) \)
7. \( \langle \text{fond} \rangle = \lambda x \lambda y. \text{fond-of}(x)(y) \)
8. \( \langle \text{fond}_2 \rangle = \lambda y \lambda f \lambda x. f(x) \cap \text{fond-of}(y)(x) \)
9. \( \langle \text{Joe} \rangle = \text{joe} \)
10. \( \langle \text{Texas} \rangle = \text{texas} \)
11. \( \langle \text{Kaline} \rangle = \text{kaline} \)
12. \( \langle \text{Lockhart} \rangle = \text{lockhart} \)

For each of the trees below, provide a fully \( \beta \)-reduced translation at each node, and state the type of the expression.

(a) \[
\begin{array}{c}
S \\
\downarrow \\
DP & VP \\
\downarrow & \downarrow \\
Joe & is & PP \\
\downarrow & \downarrow & \\
P & DP \\
\downarrow & \downarrow \\
in & Texas
\end{array}
\]
Translating to lambda calculus
Translating to lambda calculus

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

(f) S
   DP
  |  VP
Kaline  is  DP
   D  N'
  |  a
N'  AP
   A  PP
   gray2  N
   N  P  DP
   cat  in2  Texas
   AP  PP
   A  P  DP
   fond2  of  Joe
Exercise 18. Give an appropriate denotation for \( is_2 \) and derive a fully \( \beta \)-reduced translation for each node in the following trees. Treat \textit{Julius} and \textit{Amherst} as ordinary proper names.

\[
\begin{array}{c}
S \\
| \\
DP & VP \\
| \\
Julius & V & AP \\
| & | \\
is_2 & A \\
| \\
gray_2 \\
S \\
| \\
DP & VP \\
| \\
Julius & V & PP \\
| & | \\
is_2 & P & DP \\
| \\
in_2 & Amherst
\end{array}
\]

Exercise 19. \textit{Homer is a former drunkard} does not entail \textit{Homer is a drunkard} and *\textit{Homer is former}. In this sense, \textit{former} is a non-intersective modifier. Which of the following are non-intersective modifiers? Give examples to support your point.

(a) \textit{yellow}

(b) \textit{round}

(c) \textit{alleged}

(d) \textit{future}
Exercise 20. In Russian, there is a morphological alternation between two forms of adjectives, a long form and a short form. For example, the adjective ‘good’, which is feminine, is xoroša in the short form and xorošaja in the long form. Similarly, ‘talented’, which is also feminine, is talantliva in the short form and talantivaja in the long form. Masculine ‘intelligent’ is umen in the short form and umnyj in the long form.

As discussed by Siegel (1976), the two forms have different syntactic distributions. In attributive position (modifying a noun), only the long form is possible:

(18) xorošaja teorija
good-LONG theory
‘good theory’

A version with the short form, *xoroša teorija, would be ungrammatical. But in predicative positions, both forms are possible.

(19) Naša molodež’ talantlivaja i truduljubivaja
our youth talented-LONG and industrious-LONG
‘Our youth is talented and industrious’

(20) Naša molodež’ talantliva i truduljubiva
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.
5.3 Quantifiers

5.3.1 Quantifiers: not type $e$

Previously, we analyzed indefinite descriptions like a baby in sentences like Maggie is a baby. But we still cannot account for the use of a baby as the subject of a sentence or the object of a transitive verb, as in the following sentences:

(21) a. A baby loves Homer.
    b. Homer loves a baby.

We have analyzed the meaning of $a$ as an identity function on predicates, so a baby denotes a predicate (i.e., a function of type $\langle e, t \rangle$). In a sentence like A baby loves Homer, the VP loves Homer denotes a predicate of type $\langle e, t \rangle$. This leaves us in the following predicament.

```
S
   ????
  /   \
/     \ 
DP     VP
  |     |
  \    \ 
   /     / 
  D  N  V  DP
  |  |  |  |
  a baby loves Homer
```

The only composition rule that we have for combining two expressions of type $\langle e, t \rangle$ is Predicate Modification. But this yields a predicate, and it doesn’t make sense to analyze a sentence like A baby loves Homer as denoting a predicate, because the sentence is
something that can be true or false. Its semantic value in a model should be true or false, depending on whether or not a baby loves Homer.

We also lack an analysis for the following sentences.

(22) Somebody is lazy.
(23) Everybody is lazy.
(24) Nobody is lazy.
(25) \{Some, every, at least one, at most one, no\} linguist is lazy.
(26) \{Few, some, several, many, most, more than two\} linguists are lazy.

We have been assuming that the VP denotes a function of type \(\langle e, t \rangle\). A sentence is something that can be true or false, so that should be of type \(t\). So what type is the subject in these examples?

\[
S \quad t
\]

\[
DP \quad VP
\]

\[
? \quad \langle e, t \rangle
\]

\[
| \quad \text{is lazy}
\]

We could arrive at type \(t\) at the S node by treating these expressions as type \(e\), like Homer, Maggie, and the baby. But there is considerable evidence that these expressions cannot be treated as type \(e\). The analysis of these expressions as type \(e\) makes a number of predictions that are not borne out.

First, an individual-denoting should validate **subset-to-superset inferences**, for example:

(27) John came yesterday morning.
    Therefore, John came yesterday.

This is a valid inference if the subject, like John, denotes an individual. Here is why. Everything that came yesterday morning
came yesterday. If \( \alpha \) denotes an individual, then \( \alpha \) came yesterday morning is true if the individual denoted by \( \alpha \) is among the things that came yesterday morning. If that is true, then that individual is among the things that came yesterday. So if the first sentence is true, then the second sentence is true.

Some of the expressions in (22)–(26) fail to validate subset-to-superset inferences. For example:

(28) At most one letter came yesterday morning.

Therefore, at most one letter came yesterday.

This inference is not valid, so at most one letter must not denote an individual.

**Exercise 21.** Which of the expressions in (25) validate subset-to-superset inferences? Give examples.

The second property that expressions of type \( e \) should have that these expressions do not always have is that they validate the **law of contradiction.** In logic, the law of contradiction is that \( P \land \neg P \) is always false. In this context, the prediction is that sentences like the following should be self-contradictory:

(29) Mount Rainier is on this side of the border, and Mount Rainier is on the other side of the border.

This sentence is contradictory because Mount Rainier denotes an individual. Here is why. Nothing that is on this side of the border is on the other side of the border. If \( \alpha \) is of type \( e \), then \( \alpha \) is on this side of the border is true if and only if the individual that \( \alpha \) denotes is on this side of the border. This means that this individual is not on the other side of the border. So the second conjunct must be false. So the conjunction (under a standard analysis of and) can never be true.

But the following sentence is not contradictory:
More than two mountains are on this side of the border, and more than two mountains are on the other side of the border. So *more than two mountains* must not be type \( e \).

**Exercise 22.** Which of the expressions in (25) fail to validate the law of contradiction? Give examples.

**Exercise 23.** This sentence is not contradictory: *At most two mountains are on this side of the border, and at most two mountains are on the other side of the border.* This proves 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: ...”)

Finally, an expression of type \( e \) should validate the **law of the excluded middle**. The law of the excluded middle says that either \( P \) is true, or \( \neg P \) is true. It can't be the case that neither is true. For example:

(31) I am over 30 years old, or I am under 40 years old.

This is a tautology, and that is because \( I \) is an expression of type \( e \). Any expression of type \( e \) will yield a tautology in a sentence like this. Here is why. Everything is either over 30 years old or under 40 years old. If \( \alpha \) is of type \( e \), then \( \alpha \ is over 30 years old \) means that the individual that \( \alpha \) denotes is over 30 years old. \( \alpha \ is under 40 years old \) means that the individual is under 40 years old. Since everything satisfies at least one of these criteria, the disjunction (under a standard analysis of or) cannot fail to be true.

But this sentence is not a tautology:

(32) Every woman in this room is over 30 years old, or every woman in this room is under 40 years old.
So *every woman* must not be of type $e$.

**Exercise 24.** Which of the expressions in (22)–(26) fail to validate the law of the excluded middle? Give examples.

5.3.2 **Solution: Quantifiers**

Let us recap. We assume that a VP denotes a predicate (type $(e, t)$) 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 is to feed the VP as an *argument* to the *subject DP*. An DP like *something* will denote a function that takes a predicate as an argument, and returns a truth value. Its type will therefore be:

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

This is the type of a **quantifier**.

We can define *something* as a function that takes as input a predicate and returns true if and only if there is at least one satisfier of the predicate:

(33) $\llangle \text{something} \rrangle = \lambda P. \exists x. P(x)$

In contrast, the function denoted by $\llangle \text{nothing} \rrangle$ returns true if there is nothing satisfying the predicate in the relevant model:

(34) $\llangle \text{nothing} \rrangle = \lambda P. \neg \exists x. P(x)$

$\llangle \text{Everything} \rrangle$ returns true if everything satisfies the predicate:

(35) $\llangle \text{everything} \rrangle = \lambda P. \forall x P(x)$

You can think of quantifiers 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.
Using Functional Application (which, as you may recall, 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:

\[
\begin{align*}
S & \quad \text{vs.} \quad S \\
DP & \quad \text{VP} \\
\langle \langle e, t \rangle, t \rangle & \quad \langle e, t \rangle \\
\text{everything} & \quad V \\
\langle e, t \rangle & \quad \text{vanished}
\end{align*}
\]

Now what about determiners like every, no, and some? These should denote functions that take the denotation of a noun phrase and return a quantifier, because we want every girl to function in the same way as everyone. The input (e.g. girl) is type \(\langle e, t \rangle\), and the output is a quantifier, type \(\langle \langle e, t \rangle, t \rangle\). So the type of these kinds of determiners will be:

\[
\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle
\]

In particular, these determiners can be defined as follows:

\[
(36) \quad \langle\textit{some}\rangle = \lambda P \lambda Q. \exists x. [P(x) \land Q(x)]
\]

\[
(37) \quad \langle\textit{no}\rangle = \lambda P \lambda Q. \neg \exists x. [P(x) \land Q(x)]
\]

\[
(38) \quad \langle\textit{every}\rangle = \lambda P \lambda Q. \forall x. [P(x) \rightarrow Q(x)]
\]

These will yield analyses like the following:
Historical note: Treating quantifiers as type \(\langle e, t \rangle, \langle \langle e, t \rangle, t \rangle\), i.e., as relations between two sets, has roots in an influential paper by Barwise & Cooper (1981). In that paper, Barwise and Cooper argue that first-order logic is not sufficient for expressing the meanings of certain quantifiers in English including most, as first-order logic does not have a means of expressing the concept ‘more than half’. They offered a more general theory of quantificational expressions, covering the full range of so-called generalized quantifiers.

Exercise 25. Give an analysis of *A baby loves Homer* 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 \(L_\lambda\). The sentence should be true in any model where there

---

6Recent theories of words like most, building especially on work by Hackl (2009), treats most as the superlative form of words like many. This approach reopens the question of what the existence quantifiers like most signifies for linguistic theory.
is some individual that is both a baby and someone who loves Homer. You may have to introduce a new lexical entry for $a$.

**Exercise 26.** 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 type $\langle e, t \rangle$.
- Transitive verbs are type $\langle e, \langle e, t \rangle \rangle$.
- Proper names are type $e$. (You can treat *onions* as type $e$ as well.)
- Quantificational DPs are type $\langle \langle e, t \rangle, t \rangle$.
- Quantifiers are 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 onions* should be predicted to be true in a model such that no individual stands in the ‘like’ relation to onions.

(a)

```
      S
     /\  
    /  \ 
   /    \
  DP   VP
     |    |
Everybody    V
      |    |
         |    poops
```
Translating to lambda calculus
Exercise 27. 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 outdoors is not a contradiction.

(b) Everybody here is over 30 or everybody here is under 40 is not a tautology.

Exercise 28. 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 underlingly be Sam smokes or Sam drinks.

1. What translation into $L\lambda$ 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.
Exercise 29. In the Algonquian language Passamaquoddyy (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, (39) and (40) differ in voice-marking and are true in different circumstances. (The morphological glosses have been simplified.)

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

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

In (39), 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 5.1, but not in the situation on the left. (Images created by Benjamin

Figure 5.1: 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 29.)
In (40), the verb is in indirect voice, and again the agent corresponds to an indefinite noun phrase meaning ‘a man’, and the patient corresponds to a ‘all bottles’. This version of the sentence can be interpreted in two ways, one where the picture on the left in Figure 5.1 makes it true, and one where the picture on the right makes it true.

(a) Write out representations in L₁ for the two possible scope interpretations.

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

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

5.4 The definite article

So far, we have seen two types for determiners:

- ⟨⟨e, t⟩, ⟨e, t⟩⟩ for the indefinite article a in predicative descriptions such as a drunkard in Homer is a drunkard

- ⟨⟨e, t⟩, ⟨⟨e, t⟩, t⟩⟩ for quantifiers

This section presents arguments for a treatment of definite determiners with yet a third type, namely ⟨⟨e, t⟩, e⟩. In a phrase like the moon, the definite determiner the takes as input the predicate moon, and returns the unique individual who satisfies that predicate, if there is one. If there is not, then the phrase has an ‘undefined’ semantic value.
To motivate this analysis, let us first observe that definite descriptions 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).

[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’:

$$\exists x. [\text{princess}(x) \land \forall y. [\text{princess}(y) \rightarrow x = y] \land \text{smokes}(x)]$$

According to Russell’s treatment, the definite article 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 no princesses or multiple ones.

**Exercise 30.** Read the above formula aloud to yourself and then write out the words that you said. Which part of this formula ensures uniqueness?

**Exercise 31.**

(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:

```
S
   /\  \\
  DP  VP
 /    |
D     NP  V
      /  |
the  princess  smokes
```
Strawson (1950) finds this analysis very strange, on the grounds that it does not deal well with 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 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. He 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:

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 article 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

7 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.
you whether you thought that what he had just said was true, or was false; whether you agreed or disagreed with what he had just said. I think you would be inclined, with some hesitation, to say that you did not do either; that the question of whether his statement was true or false simply did not arise, because there was no such person as the King of France. You might, if he were obviously serious (had a dazed, astray-in-the-centuries look), say something like: I’m afraid you must be under a misapprehension. France is not a monarchy. There is no King of France.

Strawson’s observation is that we feel squeamish when asked to judge the 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. The sentence is neither true nor false. One way of interpreting this idea is by introducing a third truth value: Along with ‘true’ and ‘false’, we will have ‘undefined’ or ‘nonsense’ as a truth value. 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 such a sentence.

Intuitively, the reason that the sentence is neither true nor false is that there is an attempt to refer to something that does not exist. This is an intuition that was expressed earlier by Frege (1892 [reprinted 1948]). According to Frege, a definite description like this, like a proper name, denotes an individual (corresponding to type e). Regarding the negative square root of 4, Frege says:

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. [emphasis added]
We assume that by “concept-expression”, Frege means an expression of type $\langle e, t \rangle$, and that “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 meaning of type $\langle e, \langle e, t \rangle \rangle$, and that “of is vacuous, square root applies to 4 via Functional Application, and the result of that composes with negative under predicate modification.” Spelling this out yields the following structure:

```
NP
  e
  /\-
   D       N'
     \ /  \ /
      \   \ /
        \  the

A
  \ /  \ /
  \   \ /
  \   \ /
    \  negative

N
  \ /  \ /
  \   \ /
  \   \ /
    \  square root

PP
  \ /  \ /
  \   \ /
  \   \ /
    \  of

NP
  e
  /\-
   P       NP
     \ /  \ /
      \   \ /
        \  of
```

Now, what does Frege mean by “permissible”? One reasonable interpretation 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. In other words, the definite article presupposes existence and uniqueness. One might interpret this to mean that the king of France does not denote an individual if there is no King of France, even though it is an expression of type $e$. One way of handling this is to introduce a special ‘undefined individual’ of type $e$. We
will adopt this approach here, using the symbol \( \neq_e \) to denote this individual in our meta-language. Note that we will not add this symbol to our logical representation language \( L_\lambda \); rather we use \( \neq_e \) in our meta-language to refer to this ‘undefined entity’ we are imagining, specifying this as the denotation for empty descriptions.\(^8\)

In order to express Frege’s idea, we need to introduce a new symbol into the logic that we are translating English into:

\[
\iota
\]

which is the Greek letter ‘iota’. Like the \( \lambda \) symbol, \( \iota \) can bind a variable. Here is an example:

\[
\iota x. P(x)
\]

This is an expression of type \( e \) denoting the unique individual satisfying \( P \) if there is exactly one such individual, and is otherwise undefined. To add this symbol to our logic, first we add a syntax rule specifying that, given any variable \( u \) and formula \( \phi \), \( [\iota u. \phi] \) is an expression of type \( e \):

\[
(41) \quad L_\lambda \text{ syntax rule for } \iota
\]

If \( \phi \) is an expression of type \( t \), and \( u \) is a variable of type \( \sigma \), then \( \iota u. \phi \) is an expression of type \( \sigma \).

As usual, we will drop the brackets when they are unnecessary. Then we add a rule specifying its semantics:

\[
(42) \quad L_\lambda \text{ semantic rule for } \iota
\]

\[
[\iota u. \phi]^M,g = \begin{cases} 
  d & \text{if } \{ k : [\phi]^M,g[u \to k] = 1 \} = \{ d \} \\
  \neq_e & \text{otherwise}
\end{cases}
\]

\(^8\)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’. 
Exercise 32. Read the semantic rule for \( \iota \) aloud to yourself and then write down the words that you said. How does this definition ensure that \( \iota \) 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:

\[
\langle \text{the} \rangle = \lambda P. \iota x. P(x)
\]

Applied to a predicate-denoting expression like president, it denotes the unique president, if there is one and only one president in the relevant domain.

\[
\text{NP}
\]

\[
\iota x. \text{president}(x)
\]

\[
\text{D} \quad \text{N'}
\]

\[
\langle \langle e, t \rangle, e \rangle \quad \langle e, t \rangle
\]

\[
\lambda P. \iota x. P(x) \quad \text{president}
\]

\[
\text{the} \quad \text{president}
\]

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 only yields true or false for actual individuals, and yields the undefined truth value \( \# \) when given the undefined individual \( \#_e \) as input. So:

\[
\llbracket \iota x. \text{president}(x) \rrbracket^{M,g}
\]

\[
= \begin{cases} 
  d & \text{if } \{ k : [\text{president}]^{M,g[x \mapsto k]} = 1 \} = \{ d \} \\
  \#_e & \text{otherwise}
\end{cases}
\]

\[
\llbracket \text{bald}(\alpha) \rrbracket^{M,g}
\]
will be $\#$ if $\sem{\alpha}^M_{g} = \#_e$, and 1 or 0 otherwise, depending on whether $\sem{\alpha}^M_{g}$ is in the extension of bald with respect to $M$ and $g$. Since (the translation of) *the king of France* (used today) would have $\#_e$ as its denotation, (the translation of) *The King of France is bald* would then have $\#$ as its denotation.

**Exercise 33.** Explain how this Fregean treatment of the definite article vindicates Strawson’s intuitions.

**Exercise 34.** Using the assumptions above, compute a derivation for the following tree:

```
S
  /\       /\      /
DP  VP    DP  VP
   /\       /\      /
 D  NP    V  AP
  |   |    |   |
 the N PP is A
  |   |   |
  king P bald
```

This exercise can be solved using the Lambda Calculator.

**Exercise 35.** Compute a derivation for the following tree according to Frege’s intuitions, translating *square root* as a constant of type $t$, $\langle e, \langle e, t \rangle \rangle$, and *four* as a constant of type $e$:
Exercise 36. Assuming the following lexical entries:

1. $\langle book \rangle = \lambda x. \text{book}(x)$
2. $\langle on \rangle = \lambda x \lambda y. \text{on}(y, x)$
3. $\langle pillow \rangle = \lambda x. \text{pillow}(x)$

Which of the following trees gives the right kind of interpretation for the book on the pillow?
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:

(46) The opera by Mozart is beautiful.

This might have the following translation:

\[
\text{beautiful}(\lambda x. [\text{opera}(x) \land \text{by}(\text{mozart})(x)])
\]

But this expression will denote \(\not\equiv\) in a model where there are multiple operas by Mozart, assuming that beautiful yields the value \(\not\equiv\) when applied to an expression whose semantic value is \(\not\equiv\). So

---

9Here we are setting aside the fact that *beautiful* is a so-called ‘predicate of personal taste’. One might argue that possible worlds do not settle the question of which items are beautiful and which are not. If the models we are using are meant to correspond to possible worlds, then the questions of beauty are not settled relative to a model. One way out is to see models as encoding both factual information and matters of opinion, as in outlook-based semantics (Coppen, under second-round review). A number of other options are available as well, which go under the headings of ‘relativism’, ‘contextualism’, ‘absolutism’ and ‘expressivism’. 

---
again, the undefinedness of the definite description “percolates up”, as it were, to the sentence level.

**Exercise 37.** Both *The king of France is bald* and *The opera by Mozart is boring* have an undefined value relative to the actual world, but for different reasons. Explain the difference.

To summarize, we have seen a number of different types for determiners in this chapter:

- Type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$: for the lexical entry for $a$ as an identity function on predicates
- Type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$: for generalized quantifiers
- Type $\langle\langle e, t\rangle, e\rangle$: for the definite article *the*

To analyze the definite determiner, we have augmented our logical representation language with the $\iota$ symbol, which forms expressions of type $e$. The expressions denote the undefined individual if the description in question does not have exactly one satisifier. In this way, we explain Strawson’s intuition of ‘squeamishness’ regarding the question of whether the king of France exists. What we have said about the definite article does not exhaust all there is to say about presupposition; we will give a fuller discussion in Chapter 8.
6 | Variables in Natural Language

6.1 Relative clauses

In this chapter, we will consider expressions in natural language that can be translated into $L_\lambda$ as variables like $x$ and $y$. We will consider two kinds of expressions that can be treated as variables:

- pronouns like *he* and *she*
- the traces left behind by syntactic movement

Two constructions thought to involve movement traces are wh-questions like (1) and relative clauses as in (2).

1. What did Joe buy?

2. The car [which Joe bought] is very fancy.

In both of these constructions, the verb *buy* is not followed by an object, but there is an element of the sentence that intuitively corresponds to the object of the verb (*what* or *which*). One way of understanding this connection is by assuming that there are (at least) two levels of syntactic representation, one where *what*/*which* occupies the object position of the verb (‘Deep Structure’), and another where it has moved to its surface position (‘Surface Structure’). But these wh- words do not disappear entirely from their original positions; they leave a TRACE signifying that they once were there.
The structure of a question like *What did Joe buy?* after movement would be:

(3)

```
CP
  \[\text{What}_i\]
  \[C'\]
  \[C\]
  \[\text{did}\]
  \[DP\]
  \[\text{Joe}\]
  \[\text{VP}\]
  \[\text{V}\]
  \[\text{DP}\]
  \[\text{buy}\]
  \[t_i\]
```

where *i* is an arbitrary natural number. The subscript *i* on *what* is an INDEX, which allows us to link the *wh*-word to its base position. It can be instantiated as any natural number (1, 2, 3, ...). The element *t_i* is a trace of movement, co-indexed with the moved relative pronoun. 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’*).\(^1\)

The relative clause in (2) can be analyzed similarly:

\(^1\)The term ‘specifier’ comes from the *X*-bar theory of syntax, where all phrases are of the form \([_{X_P} \text{ (specifier)} [_{X'} [_{X} \text{ (complement)} ] ]]\).
This structure is derived through a syntactic transformation called RELATIVIZATION. In the structure in (4), the relative pronoun which occupies the specifier position of CP (sister to C'). In this structure, the C position is thought to be occupied by a silent version of the complementizer that; more on that below. As the wh-word and the trace bear the same index, we say that the two expressions are CO-INDEXED. The presence of a trace in a structure signifies that at some deeper level of representation, the relative pronoun which occupied the object position of the verb. This assumption allows the wh-element to be interpreted as the object of the verb, even though it does not occupy the object position at the level where it is pronounced.

A relative clause like who likes Joe, as in the girl who likes Joe, in which the subject is extracted, will have the trace in subject position thus:
In this tree, the relative pronoun who is co-indexed with a trace in the subject position for the embedded verb likes.

The silent complementizer that in (4) in (5) is pronounced in a relative clause like that Joe bought in a phrase like car that Joe bought. In such a case, it is the complementizer that instead of the relative pronoun which that we hear, but there is still assumed to be a relative pronoun that undergoes movement.

So at some deeper level of representation, a relative clause like car that Joe bought would be pronounced car which that Joe bought, with the relative pronoun which preceding the complementizer that. One reason to think that the word that is not of the same
category as a relative pronoun as *which* is that only relative pronouns participate in so-called ‘pied-piping’:

(7) good old-fashioned values [\_\_CP on *which* we used to rely ]

Compare: *...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.*

In some languages, including Bavarian German, in which relative pronouns can actually co-occur with complementizers (Carnie 2013, Ch. 12).

(8) I woass ned **wann dass** da Xavea kummt.
I know not **when that** the Xavea comes
‘I don’t know when Xavea is coming’

This provides additional evidence for the idea that relative pronouns like *which* 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.\(^2\)

\(^2\)See Carnie (2013, Ch. 12) for a more thorough introduction to the syntax of relative clauses.

These syntactic assumptions lay the groundwork for a satisfactory semantics of relative clauses. Semantically, relative clauses are much like adjectival modifiers; compare:

(9) The **red** car is very fancy.

(10) The car **that Joe bought** is very fancy.
We can capture the semantic contribution of relative clauses by assuming that they denote predicates and can combine via Predicate Modification.

(11) \[
\text{NP} \quad \langle e, t \rangle \\
\quad \text{NP} \quad \langle e, t \rangle \\
\quad \text{CP} \quad \langle e, t \rangle \\
\quad \text{car} \quad \text{that Joe bought}
\]

Intuitively, what a relative clause like *which Joe bought* denotes is a predicate which holds of a given entity \( x \) if and only if Joe bought \( x \). We can represent such a property with the following lambda expression:

(12) \[\lambda x. \text{bought}(\text{joe}, x)\]

Then, the restrictive relative clause *car which Joe bought* obtains the following meaning through Predicate Modification:

(13) \[\lambda x. \text{car}(x) \land \text{bought}(\text{joe}, x)\]

This is the goal. This goal can be reached by making the following assumptions:

- Relative clauses are formed through a movement operation that leaves a trace.
- Traces correspond to logical 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 \( L_\lambda \) we have an infinite number of constants and variables in stock. For every natural number \( i \) and every type \( \tau \), we have a constant of the form \( c_{i,\tau} \).
and a variable of the form

\[ v_{i, \tau} \]

Assuming that traces always correspond to variables of type \( e \), then it is natural to assume that, for example, the trace \( t_7 \) is interpreted as the variable \( v_{7, e} \):

\[ \langle t_7 \rangle = v_{7, e} \]

The denotation of the variable \( v_{7, e} \) will depend on an assignment; recall that \[ \langle v_{7, e} \rangle^{M, g} = g(v_{7, e}) \]

Not only is it natural to assume that the index of the trace matches the index of the variable that it is mapped to; it is also convenient to do so, as we will see later: Mapping a trace with index \( i \) to a variable with the same index puts us in a position to choose a matching variable for the lambda expression to bind when reach the co-indexed relative pronoun in the tree.

We have thus arrived at a new composition rule:

**Composition Rule 4. Pronouns and Traces Rule**

If \( \alpha \) is an indexed trace or pronoun, \[ \langle \alpha \rangle = v_{i, e} \]

---

3 Contrast Heim and Kratzer’s rule, given in a direct interpretation style, where an assignment function decorates the denotation brackets: \[ \langle \alpha \rangle^S = g(i) \]. Here the difference between direct and indirect interpretation becomes bigger than mere substitution of square brackets for angled brackets: In indirect interpretation, we translate natural language variables as logical variables, rather than as the value of assignment functions. 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 \[ \langle \text{loves him}_i \rangle^S = \lambda x. \text{loves} \ g(i) \]. Here, variables like \( x \) appear on the righthand side and variables like ‘him’ appear on the lefthand side. Thus in Heim and Kratzer’s style, assignment functions must play a role at two levels: at the translation from English to logic, and in the interpretation of the logic. The distinction between the assignment functions at these two levels is rarely spelled out, if ever. With indirect interpretation, we only need assignment functions at one level (the logical level).
We have called it the ‘Pronouns and Traces Rule’ because it will also be used for pronouns; for example:

$$\langle he \rangle = v_{7,e}$$

More on the pronoun side of this in §6.4.

Note: We will sometimes leave off $e$ from the subscripts on variables when it is clear what the type is. So we may abbreviate $v_{7,e}$ as $v_7$.

**Exercise 1.** Give translations at every node for the following tree:

```
  S
  /\  \\
DP   VP
/   |   |
He3  V   DP
   |     |
bought  it5
```

With these assumptions, we derive the representation

$$\text{bought}(j, v_1)$$

for *Joe bought* $t_1$:

---

4 The idea of treating traces and pronouns as variables is rather controversial; see [Jacobson 1999](#) and [Jacobson 2000](#) for critique and alternatives.
The translation corresponding to this S node, \( \text{bought}(j, v_1) \), is of type \( t \). Suppose that the complementizer \( \text{that} \) is a trivial identity function of type \( \langle t, t \rangle \), so \( \langle \text{that} \rangle = \lambda p. p \), where \( p \) is a variable of type \( t \). So \( \text{that} \ Joe \ bought \ t_1 \) has the same translation, of type \( \langle e, t \rangle \)? In particular, how do we reach our goal, according to which the relative clause ends up with a translation equivalent to \( \lambda x. \text{bought}(j, x) \)?

We can achieve our goal by assigning the relative clause an interpretation in which a lambda operator binds the variable \( v_1 \), thus:

\[
\lambda v_1. \text{bought}(x, v_1)
\]

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, triggered by the presence of an indexed relative pronoun, lambda-abstracts over the appropriate variable:
Composition Rule 5. Predicate Abstraction
If $\gamma$ is an expression whose only two subtrees are $\alpha_i$ and $\beta$ and $\langle \beta \rangle$ is an expression of type $t$, then $\langle \gamma \rangle = \lambda v_i.e \cdot \langle \beta \rangle$.

This gives us the following analysis of the relative clause:

(14)

We have reached our goal! The relative clause that Joe bought denotes the property of having been bought by Joe. This can combine via Predicate Modification with car, giving the property of being a car that Joe bought, as it should.
Exercise 2. (a) For each of the labelled nodes in the following tree, give: i) the type; ii) a fully $\beta$-reduced translation to $L_\lambda$, and iii) the composition rule that is used at the node.

(b) Notice that you are not asked to give a type for $who_1$. Why not? Hint: Use the word 'syncategorematically'.

Exercise 3. 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. John, who I like, is coming to the party.
2. *John who I like is coming to the party.
3. That guy, who I like, is coming to the party.
4. The guy 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 4.** For each node in the following tree, give the type and a fully $\beta$-reduced translation to $L_\lambda$. 

```
NP
  | NP
  | man
  | CP
  | who$_1$
  | S
  | D$_1$
  | talked
  | PP
  | P
  | to
  | D
  | the
  | NP
  | boy
  | who$_2$
  | S
  | D$_2$
  | V
  | visited
  | DP
  | him$_1$
```
6.2 Quantifiers in object position

6.2.1 Quantifier Raising

The rule of Predicate Abstraction will also help with the interpretation of quantifiers. Recall that a sentence like every linguist offended John, with a quantificational noun phrase in subject position, receives an analysis like this:

(15)

But a problem arises when the quantifier is in object position. If we try to interpret the following tree, we get a type-mismatch:
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! We could represent the meaning as follows:

\[
(17) \quad \forall x[\text{linguist}(x) \rightarrow \text{offended}(\text{john}, x)]
\]

According to the assumptions we made so far, *every linguist* translates as:

\[
(18) \quad \lambda Q \forall x[\text{linguist}(x) \rightarrow Q(x)]
\]

The appropriate value for \( Q \) here would be a function that holds of \( x \) if John offended \( x \), representable by:

\[
(19) \quad \lambda x. \text{offended}(\text{john}, x)
\]

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 meaning:

\[
(20) \quad [\lambda Q \forall x[\text{linguist}(x) \rightarrow Q(x)]](\lambda x.\text{offended}(\text{john}, x)) \\
= \forall x[\text{linguist}(x) \rightarrow \text{offended}(\text{john}, x)]
\]

**Exercise 5.** Simplify the following expression step-by-step:

\[
[\lambda Q. \forall x[\text{linguist}(x) \rightarrow Q(x)]](\lambda v_1.\text{offended}(\text{john}, v_1))
\]

Tip: Use the ‘scratch pad’ function in the Lambda Calculator.

We can get the components we need to produce the right meaning using the rule of Quantifier Raising. Quantifier raising is a syntactic transformation moving a quantifier (an expression of type \(\langle \langle e, t \rangle, t \rangle\)) to a position in the tree where it can be interpreted, leaving a DP trace in object position. This transformation occurs not between Deep Structure and Surface Structure, but rather between Surface Structure and another level of representation called Logical Form. 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 (21a) is converted to the Logical Form representation (21b):

\[
(21) \quad \text{a.}
\]
The number 1 in the syntax tree plays the same role as the 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‘; Functional Application is used at all other branching nodes.
The Quantifier Raising solution to the problem of quantifiers in object position is embedded in a syntactic theory with several levels of representation:

- **Deep Structure (DS):** Where the derivation begins, and 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)\(^5\)

Transformations map from DS to SS, and from SS to PF and LF:

\[
\begin{array}{c}
\text{DS} \\
\mid \\
\text{SS} \\
\text{LF} \\
\text{PF}
\end{array}
\]

This is the so-called ‘T-model’, or (inverted) ‘Y-model’ of Government and Binding theory, motivated originally by Wasow (1972) and Chomsky (1973). Since the transformations from SS to LF happen “after” the order of the words is determined, we do not see the output of these transformations. These movement operations are in this sense **COVERT**.

Many other transformational generative theories of grammar have been proposed over the years, of course (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. Of course, 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.

---

\(^5\)Note that ‘Logical Form’ refers here to a **level of syntactic representation**. A Logical Form is thus a natural language expression, which will be translated into \(L_\Lambda\). It is natural to refer to the \(L_\Lambda\) translation as the ‘logical form’ of a sentence, but this is not what is meant by ‘Logical Form’ in this context.
**Exercise 6.** Derive a translation into 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 also 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 7.** *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 both readings, and specify the translation into \( L_\Lambda \) at every node of your trees. You can do this in the Lambda Calculator.

**Exercise 8.** 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 Chomsky.
3. Chomsky respects nobody who smokes.
4. Chomsky 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 $L_\lambda$)

• composition rules (Functional 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 meaning in $L_\lambda$
  – 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.

6.2.2 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 imagine type-shifting rules that have the power to repair the mismatch. These could 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’ 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 9.** 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 the expression to a more complex type). We will focus on one instantiation of this schema, called OBJECT RAISING, defined as follows.
**Type-Shifting Rule 2. Object raising** \((\text{RAISE})\)

If \(\llbracket \alpha \rrbracket\) is of type \(\langle e, t \rangle\) then:

\[
\llbracket \text{RAISE } \alpha \rrbracket = \lambda v_{\langle (e, t), t \rangle} \lambda y. v(\lambda z. \llbracket \alpha \rrbracket(z)(y))
\]

(unless \(v, y\) or \(z\) is free in \(\llbracket \alpha \rrbracket\); in that case, use different variables).

So a sentence like *Sam loves everybody* will be analyzed as follows:

(23)

![Diagram of the analysis of the sentence *Sam loves everybody*.](image)

**Exercise 10.** Reproduce the tree in (23) 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*. 
Note that this is not all that Hendriks’s system can do. He defines a general schema of which the Object Raising rule is a particular instance. The general schema is as follows: If an expression has a translation $\alpha$ of type $\langle \overrightarrow{a}, \langle b, \langle \overrightarrow{c} \rightarrow t \rangle \rangle \rangle$, where $\overrightarrow{a}$ is a possibly null sequence of types, then that expression also has translations of the following form:

$$\lambda \overrightarrow{x}^{\overrightarrow{a}} \lambda \nu_{\langle (b, t), t \rangle} \lambda \overrightarrow{y}^{\overrightarrow{c}} \nu(\lambda z_{b}[\alpha(\overrightarrow{x})(z)(\overrightarrow{y})])$$

A sentence with two quantifiers like Everybody likes somebody can be analyzed in two different ways, depending on the order in which the argument slots of the verb are lifted. The resulting type is the same: something that combines with two quantifiers. But the different orders in which the type-shifting operations applied give two different scopes.

It turns out that if you raise the object first and then the subject, then you get a reading with the existential quantifier outscoping the universal quantifier. This is shown in the following tree, where subscripts indicate the types of the variables.

---

6 with $\overrightarrow{a}$ as $e$, $\overrightarrow{c}$ as null, and $b$ as $e$

7 with $\overrightarrow{a}$ as $\langle \langle e, e, t \rangle \rangle$, $\overrightarrow{c}$ as null, and $b$ as $e$
If the subject is raised first\footnote{with $\vec{a}$ as $e$, $\vec{c}$ as null, and $b$ as $e$} and then the object\footnote{with $\vec{a}$ as null, $\vec{c}$ as $\langle e, t \rangle$, and $b$ as $e$} then the universal quantifier ends up scoping outside of the existential quantifier.
6.2.3 Putative arguments for the movement

Heim & Kratzer (1998) give a number of arguments in favor of the QR approach.

Argument #1: Scope ambiguities. The QR approach delivers two different readings for scopally ambiguous sentences like Everybody loves somebody. Heim & Kratzer (1998) claim that this cannot be done with a flexible types approach. But they only consider flexible types approaches in which the quantifiers themselves undergo type-shifting operations; they do not consider the approach of Hendriks (1993), which does give both readings, as just shown in the previous section.

Argument #2: Inverse linking. A somewhat more challenging case for a flexible types approach falls under the heading of ‘in-
verse linking’, also known as ‘binding out of DP’. Here is a classic example:

(26) One apple in every basket is rotten.

This does not mean: ‘One apple that is in every basket is rotten’, and most flexible types analyses, including Hendriks [1993] deliver only that reading. The QR analysis gives the right reading starting from the following LF:

(27)

```
(27)  

S
   /\   /\  S
  DP 1  VP
     /\   \
  every basket     is rotten
              /\  /
          DP  NP
              /\  /
          D   NP
              /\  /
        one  apple
              /\  /
            P   PP
                /\  /
              t1  in
```

However, Barker [2005] has a flexible types analysis of binding out of DP, so this does not constitute a knock-down argument in favor of QR.

**Exercise 11.** Derive the translation for (27) compositionally using the rules we have introduced so far. You may need to introduce new lexical entries.
Exercise 12. Explain why the correct reading of (26) cannot be derived via the Object Raising type-shifting rule. Start by applying the Object Raising type-shifting rule to the preposition \textit{in}.

Argument #3: Antecedent-contained deletion Another argument that has been made in favor of QR is based on examples like this:

(28) Mary read every novel that John did.

This example involves a kind of VP ellipsis. VP ellipsis generally involves deletion of a VP under identity with an antecedent. For example, in the following case the deleted VP seems to be ‘read \textit{War and Peace}:

(29) I read \textit{War and Peace} before you did.

so the underlying structure would be:

(30) I read \textit{War and Peace} before you did read \textit{War and Peace}.

But what happens if we try to fill in the antecedent VP in a case like (28)? The VP \textit{read every novel that John did} contains the ellipsis site. We might get an infinite regress if we tried to fill it in: \textit{Mary read every novel that John did read every novel that John did read every novel that...} Since the antecedent in this case seems to contain the VP that is deleted, the phenomenon in (28) is known as \textbf{ANTECEDENT-CONTAINED DELETION}.

But there is a solution! If the quantifier phrase in (28) undergoes QR, then the antecedent VP no longer contains the deleted VP:
The antecedent VP is now identical to the elided VP except for the index on the trace. So, thanks to QR, examples like (28) can be accommodated.

**Exercise 13.** Give an analysis of (31) with fully $\beta$-reduced $\lambda$-expressions at each node. Use of the Lambda Calculator is encouraged.

**Exercise 14.** Explain how the phenomenon of antecedent-contained deletion provides an argument in favor of QR.
Note that Jacobson (1999) argues that this phenomenon does not constitute a clear argument in favor of QR. She proposes a way of handling it using a mode of composition called function composition, which we have not introduced here but has a range of interesting applications.

**Argument #4: Quantifiers that bind pronouns** Another argument that Heim and Kratzer give in favor of QR has to do with reflexive pronouns. When a reflexive pronoun is anaphoric to a proper name, the sentence can be paraphrased more or less by replacing the pronoun with its antecedent:

(32) a. Mary blamed herself.
    b. Mary blamed Mary.

But this is not the case when a reflexive pronoun is anaphoric to a quantifier:

(33) a. Every woman blamed herself.
    b. Every woman blamed every woman.

The problem is not unique to reflexive pronouns; it has to do with any use of a pronoun that is connected to a quantifier including the following:

(34) No man noticed the snake next to him.

If we treat pronouns as variables and use QR, we can easily account for the meanings of these sentences:
Suppose we have that the VP *blamed herself* has the following translation (more on this in Section 6.4):

\[(36) \quad \lambda y. \text{blamed}(y, v_1)\]

The translation of corresponding to the tree in (35) then gives an appropriate translation:

\[(37) \quad \forall x [\text{woman}(x) \rightarrow \text{blamed}(x, x)]\]

**Exercise 15.** Assuming that *herself* \(_1\) is translated as \(v_1\), draw an LF tree for *Every woman blamed herself* assuming quantifier raising and give a derivation showing how the meaning in (37) is derived.

Assuming that pronouns are translated as variables, it is somewhat more difficult to imagine how to derive the right reading on a flexible types approach. If we combine this with *every woman*, then we will end up with a formula that contains a free variable:

\[\forall x. [\text{woman}(x) \rightarrow \text{blamed}(x, v_1)]\]

where \(v_1\) is free. This is not quite the right meaning.
But there are a number of ways of dealing with reflexive pronouns without QR. For example, a reflexive pronoun can be analyzed as a function that takes as an argument a relation of type \( \langle e, \langle e, t \rangle \rangle \) and returns a reflexive predicate (Keenan, 1987; Szabolcsi, 1987):

\[
\langle \text{herself} \rangle = \lambda R \langle e, \langle e, t \rangle \rangle \lambda x. R(x, x)
\]

Applied to, for example, blamed, this will yield the property of blaming oneself, which can be combined with every woman to give the right reading.

**Exercise 16.** Use the treatment of herself in (38) to derive a translation for every woman blamed herself.

**Exercise 17.** Does the lexical entry for herself in (38) give the right results for a ditransitive case, like Mary introduced herself to Sue? Why or why not?

**Argument #5: The Extraction-Scope Generalization** Finally, it has been pointed out that there seem to be some syntactic constraints on QR, and these constraints seem to mirror constraints on movement in general. For example, the quantifier every country can take wide scope in (39a) but not in (39b); the latter has only the implausible reading implying the existence of an “international girl”.

\[
\begin{align*}
(39) & \quad \text{a. John knows a girl from every country.} \\
& \quad \text{b. #John knows a girl who is from every country.}
\end{align*}
\]

Similarly, extraction of which country is possible from a prepositional phrase modifier but not a relative clause modifier:

\[
\begin{align*}
(40) & \quad \text{a. Which country does John know a girl from?}
\end{align*}
\]
b. *Which country does John know a girl who is from?

Example (40b) is a violation of Ross’s (1968) “Complex Noun Phrase Constraint”, one of the so-called island constraints specifying islands for extraction (syntactic environments from which extraction is impossible). The oddness of (39b) might lead one to suspect that the constraints on scope are parallel to the constraints on wh-extraction.

Another parallel has to do with coordination. A wide scope reading for every man is possible in (41a), but not in (41b), where the VP is coordinated with another VP. In (41b), we have only the implausible reading that every man has the same mother.

(41) a. Some woman gave birth to every man.

b. #Some woman gave birth to every man and will eventually die.

Similarly, extraction of whom is possible in (42a) but not (42b), where there is coordination.

(42) a. I wonder whom Mary gave birth to.

b. *I wonder whom Mary gave birth to and will eventually die.

The badness of (42b) falls under Ross’s (1968) “Coordinate Structure Constraint”. It appears that quantifiers are also subject to some kind of coordinate structure constraint.

Winter (2001a) refers to this parallel between scope and wh-extraction as the “Extraction Scope Generalization”. This correlation might suggest that scope readings are generated via the same kind of movement that puts wh-phrases in their place. However, scope does not correlate completely perfectly with extraction, and QR is not the only possible explanation for the parallels that do exist.

**Summary.** While there are some phenomena that might seem prima facie to favor a QR approach, none of them constitutes a
knock-down argument in favor of it. Some may find QR easier to understand and work with and prefer it on those grounds; others may find direct compositionality more appealing. QR is certainly an important idea to have a good grasp on, as it figures in so much writing in semantics. But whether QR has any solid empirical advantages or disadvantages seems to be an open question.

6.3 Possessives

Let us now consider an application combining uniqueness presuppositions with quantification. It is often suggested that possessive descriptions contain a hidden the. One prima facie reason for believing this is that possessive descriptions with definite possessors in argument position typically behave like (constant) terms, i.e., behave as if they have denotations of type e. For example, Mary's rabbit gives rise to a contradiction in (43) whereas some rabbit does not. This would be expected if Mary's rabbit denoted a single rabbit.

(43)  

a. Mary's rabbit is in the cage and Mary's rabbit is outside the cage. (contradictory)  
b. Some rabbit is in the cage and some rabbit is outside the cage. (not contradictory)

Mary's rabbit also behaves like an individual-denoting expression when it comes to conjunction.

(44)  

a. Mary's rabbit is from South Africa and Mary's rabbit is a secret agent.  
\[\iff\] Mary's rabbit is a secret agent from South Africa.  
b. Some rabbit is from South Africa and some rabbit is a secret agent.  
\[\iff\] Some rabbit is a secret agent from South Africa.

However, as Löbner (2011) discusses, these tests do not always indicate type e status when the possessor is not definite. He points
out that *at least one child’s pet rabbit* does not behave like it refers to a particular individual for the purposes of conjunction, as shown in (45), or negation, as shown in (46).

(45) At least one child’s pet rabbit is from South Africa and at least one child’s pet rabbit is a secret agent.
\[ \iff \text{At least one child’s pet rabbit is a secret agent from South Africa.} \]

(46) At least one child’s pet rabbit isn’t in the cage, and at least one child’s rabbit is in the cage. (not contradictory)

Yet, there still appears to be a kind of uniqueness presupposition. As Kamp (2001) observes, *at least one child’s pet rabbit* is felicitous only when at least one child in the domain of discourse has exactly one pet rabbit. With the help of Quantifier Raising, we can make sense of the idea that *at least one child’s rabbit* is an expression of type $e$, even though it doesn’t refer to a particular rabbit. In other words, we can coherently maintain that possessives are inherently definite in the face of data like this.

In order to analyze these cases, we need to have an analysis of possessives. It is a bit tricky, because in a case like *John’s sister*, the possessive relation seems to come from the noun, whereas in *John’s car*, the possessive relation comes from context; John could own the car, or have been assigned to the car, etc. Nouns like *sister* are called RELATIONAL NOUNS, and are typically analyzed as being of type $\langle e, (e, t) \rangle$. So *sister* would be translated as a two-place predicate. One way to distinguish between relational and non-relational nouns is with *of*-PP complements as in (47a) vs. the so-called DOUBLE GENITIVE construction shown in (47b).

(47) a. a portrait of Picasso

b. a portrait of Picasso’s

The example without possessive ’s in (47a) describes a portrait in which Picasso is depicted. In (47b), Picasso could have made the portrait, or he could own it, or any number of other things. The
restricted interpretation of (47a) can be explained based on the following assumptions. There is one sense of \textit{portrait} on which it denotes a binary predicate, and on that sense, the argument to the noun is the thing depicted in the portrait. On that sense, \textit{portrait} is a transitive noun, and requires an object, just as transitive verbs require an object, and the of-PP construction in (47a) serves to feed the object to the noun. So the of-PP construction in (47a) is not a possessive construction at all; it is simply a construction in which a transitive noun takes an argument. There is another sense of \textit{portrait} which is a simple unary predicate, and that is the sense that surfaces in (47b), where the possessive \textit{'s} invokes a contextually salient possessive relation.

To deal with this, we will take inspiration from Vikner & Jensen’s (2002) idea that in cases like \textit{John’s car}, an \langle e, t \rangle-type noun undergoes a type shift to become an \langle e, \langle e, t \rangle \rangle-type noun whose meaning introduces an underspecified possessive relation poss. The type-shift is defined as follows:

\begin{center}
\textbf{Type-Shifting Rule 3. Possessive shift} \\
If $\llangle \alpha \rrangle$ is of type $\langle e, t \rangle$, then:

$$\llangle \text{poss } \alpha \rrangle = \lambda y \lambda x. [\llangle \alpha \rrangle(x) \land \text{poss}(x, y)]$$

as well (unless $y$ or $x$ is free in $\llangle \alpha \rrangle$; in that case, use different variables of the same type.

For example, this type-shift applied to \textit{car} will produce

$$\lambda y \lambda x. [\text{car}(x) \land \text{poss}(x, y)]$$

so the possessive shift turns \textit{car} into \textit{car of}, so to speak.

One of the virtues of the possessive shift is that it makes it possible to account for the ambiguity of examples involving former such as the following:
Variables in Natural Language

(48)  John's former mansion
      ‘the mansion that John formerly owned’
      ‘the former mansion that John currently owns’

If the possessive shift occurs between *former* and *mansion*, then we get one reading, but if it occurs above *former mansion*, then we get another reading. A full analysis of this particular example would require us to be able to talk about time, which we have so far ignored, so let us move on without getting into detail about that.

We will also assume that possessive ‘s is an identity function on binary predicates, so that any noun that it combines with has to undergo the possessive shift. \(^{10}\)

(49)  \(\langle \textsc{s}\rangle = \lambda \text{R}. \text{R}\)

(Recall that \(R\) is a variable of type \(\langle e, \langle e, t \rangle \rangle\).) The possessive ‘s will form a constituent with the modified noun. This may strike the reader as odd, but it is not so strange in light of other constructions involving clitics. (Clitics are things that are in-between a morpheme and a word. For example, in *I'm tired*, it is not unreasonable to say that 'm forms a syntactic constituent with *tired*.\(^{11}\) So the types for *John's rabbit* will work out as follows:


\(^{11}\) Barker (1995) assumes that the possessive construction involves a silent head that combines with the following noun and serves as “glue” for the possessive construction; this analysis assumes that the possessive ‘s sits in the position where Barker has a silent head.
To get a type $e$ interpretation of John’s rabbit, we can assume that there is a silent definite article in the syntax, or we can assume that John’s rabbit undergoes a type-shifting operation that does the same job. In fact, Partee (1986) already posited a type-shifting operation that would do this job, called iota.

Type-Shifting Rule 4. Iota shift
If $\langle \alpha \rangle$ is of type $\langle e, t \rangle$, then

$$\langle \text{IOTA } \alpha \rangle = \iota x. \langle \alpha \rangle(x)$$

as well (unless $x$ is free in $\alpha'$; then choose a different variable).

If we assume that this type-shifting operation applies to John’s rabbit, then we will end up with the following translation for the phrase:

$$(50) \quad \iota x. [\text{poss}(j, x) \land \text{rabit}(x)]$$
Exercise 18. Give a tree analysis of John’s rabbit showing logical representations at each node. At each node, indicate which composition rules and type-shifting operations you use, if any.

Now let us consider how to analyze a case with a quantified possessor. We can do it with Quantifier Raising. The possessor, which is a quantifier, undergoes QR, leaving a trace in its original position. The possessive noun undergoes the iota shift, yielding, in effect, a definite description containing a variable bound by the possessor. The syntactic structure and types will be as follows:

12 This is an instance of the ‘binding out of DP’ problem, so QR is making our lives easier here. See Barker (2005) and Francez (2011) regarding directly compositional analyses of quantified possessives.
Exercise 19. Derive the translation for *Some child’s pet rabbit is a secret agent* step-by-step.

Exercise 20. Consider examples (45) and (46) again in light of the foregoing discussion. Is it possible to account for them while still assuming that the possessive DP *at least one child’s pet rabbit* is of type $e$? Explain why or why not.
6.4 Pronouns

What is the meaning of a pronoun like *he*? If I point to Jared Kushner, and say:

(52) He is suspicious.

then *he* refers to Jared Kushner. But I could point to Jeff Sessions and say the same thing, in which case *he* would refer to Jeff Sessions. I don't have to point, of course; if Jeff Sessions is on TV then he is sufficiently salient for the same utterance to pick him out. Alternatively, I could raise Jeff Sessions to salience by talking about him:

(53) Jeff Sessions lied under oath about his contacts with Russia. He 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.

(Note that 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

(54) No woman blamed herself.
Neither man thought he was at fault.

Every boy loves his mother.

the book such\textsubscript{1} that Mary reviewed it\textsubscript{1}

So not all pronouns are referential. Note that it is sometimes said that *No woman* and *herself* are “coreferential” in (54) but this is strictly speaking a misuse of the term “corefrential”, because, as Heim and Kratzer point out, *coreference implies reference*.

**Exercise 21.** 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 (54)-(57) can be analyzed as bound variables. We saw a preview of this above in connection with example (33a) (*Every woman blamed herself*), which was analyzed above as:

\[(58) \langle \text{blamed herself} \rangle = \lambda y. \text{blamed}(y, v_1)\]

If we combine this with *every woman*, then we will end up with a formula that contains a free variable:

\[\forall x. [\text{woman}(x) \rightarrow \text{blamed}(y, v_1)]\]

and this is not the right reading.

Following Heim and Kratzer, *such*-relatives 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 (57), there is coindexation between *such* and *it*. The analysis works as follows:

\[\text{let us assume that } x \text{ is a distinct variable from } v_1.\]
Exercise 22. Give the types and a fully $\beta$-reduced logical translation for every node of the following tree.
Let us consider the possibility that pronouns should *always* be treated as bound variables.¹⁴

**Hypothesis 2.** All pronouns are translated as bound variables.

¹⁴Heim & 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.
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, Predicate 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 (50), we would have to QR Jeff Sessions to a position where it QRs He in the second sentence somehow. It is not completely crazy to imagine that proper names can undergo QR, but it is counterintuitive that QR should 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.

(60) 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 strict reading (the sympathetic reading), Peter blames himself. On the sloppy reading (the asshole reading), Peter blames Egon.

The strict/sloppy ambiguity exemplified in (60) 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:

\[
S \\
\text{DP} \quad \text{LP} \\
\text{I} \quad 1 \\
S \\
\text{DP} \quad \text{VP} \\
t_1 \quad \text{V} \quad \text{DP} \\
\text{blame} \quad \text{myself}_{1}
\]

The strict reading can be derived from an antecedent without QR:

\[
S \\
\text{DP} \quad \text{VP} \\
\text{I} \quad \text{V} \quad \text{DP} \\
\text{blame} \quad \text{myself}_{1}
\]

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 23. 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:

(63)
```
S
  DP  VP
    She₁  V  A
    |    |   |
    is  nice
```

(64)
```
S
  DP  VP
    John  V  DP
    |    |    |
    hates  D  NP
    |    |    |
    his₁  father
```

But in these examples, the pronoun is bound:
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 24.** 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 the next chapter.
7 Dynamic semantics

7.1 Pronouns with indefinite antecedents

The way of treating pronouns just introduced in the previous chapter brings us very close to DYNAMIC SEMANTICS, where the meaning of an utterance is something that depends on and updates the current discourse context. 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:

(1) John 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 (1982) discusses. First, it correctly captures the fact that (2) does not imply that there is a dog that John and Mary are both friends with a dog.

(2) 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 (3) does not say regarding some particular dog that it came in, in contrast to (4), which has a proper name referring to a dog and does have that implication.

(3) It is not the case that a dog came in.

(4) It is not the case that Fido came in.

Thirdly, Russell’s analysis correctly captures the fact that (5) can be true even if it is not the case that there is some particular dog that everybody owns, while (6) does not have that implication.

(5) Every child owns a dog.

(6) Every child owns Fido.

If *a dog* referred to a dog then (5) should mean that every child owns that dog, as in (6).

However, there are some problems. If we analyze example (1) 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):

(7) \( \exists x\left[ \text{cat}(x) \land \text{found}(j, x) \right] \land \text{ran-away}(v_3) \)

with \( v_3 \) an unbound variable outside the scope of the existential quantifier. (It doesn’t matter which variable we choose; even if we choose \( x \), the variable will still be unbound, because it will be outside the scope of the existential quantifier.) Assuming that QR does not move quantifiers beyond the sentence level, the scope of the existential quantifier introduced by *a cat* does not extend all the way to include the variable \( v_3 \), and there is no other variable-binder to bind it.
Exercise 1. Give LF trees and derivations for the two sentences in (1). (Feel free to treat ran away as a single verb.) Explain why these representations do not capture the connection between 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:

(8) \[\exists x [\text{cat}(x) \land \text{found}(j, x) \land \text{ran-away}(x)]\]

Regarding this imaginable solution, Heim (1982, 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 (1982) also presents a number of empirical arguments against this kind of treatment. One comes from dialogues like the following:

(9) a. A man fell over the edge.
    b. He didn’t fall; he jumped.

(10) a. A dog came in.
    b. What did it do next?

What would a Geachian analysis be for a case like [9]? If we let the existential quantifier take scope over the entire discourse, we
would get the meaning ‘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 10 presents a similarly puzzling challenge.

Another argument that Heim makes against the Geachian analysis is based on the following example:

(11) John owns some sheep. Harry vaccinated them.

This sentence should be false in a situation where John owns six sheep, of which Harry vaccinated three. On the Geachian analysis, the interpretation would be something along the lines, ‘there exists an \( x \) such that \( x \) is a bunch of sheep and John owns \( x \) and Harry vaccinated \( x \)’, which would be true in such a situation. But the English sentence would not be, so this is not a welcome prediction.

Thirdly, Geach’s proposal would mean that existential quantifiers have different scope properties from other quantifiers. Consider the following examples:

(12) A dog came in. It lay down under the table.

(13) Every dog came in. #It lay down under the table.

(14) No dog came in. #It lay down under the table.

In neither (13) nor (14) can \( it \) be bound by the quantifier in the first sentence.\(^2\) Heim (1992, 17) concludes:

\(^2\)There is a phenomenon called telescoping, counterexemplifying the generalization that every cannot take scope beyond the sentence boundary. Examples 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”)
The generalization behind this fact is that an unembedded sentence is always a “scope-island,” i.e. a unit such that no quantifier inside it can take scope beyond it. This generalization (which is just a special case of the structural restrictions on quantifier-scope and pronoun-binding that have been studied in the linguistic literature) is only true as long as the putative cases of pronouns bound by existential quantifiers under Geach’s analysis are left out of consideration.

Thus it seems that Geach’s solution will not do, and we need another alternative.

So-called ‘donkey anaphora’ is another type of case involving pronouns with indefinite antecedents that motivates dynamic semantics. The classic ‘donkey sentence’ is:

(15) If a farmer owns a donkey, then he beats it.

This example is naturally interpreted as a universal statement, representable as follows:

(16) \( \forall x[[\text{farmer}(x) \land \text{donkey}(x) \land \text{own}(x, y)] \rightarrow \text{beats}(x, y)] \)

But the representation that we would derive for it using the assumptions that we have built up so far would be:

(17) \( \exists x \exists y[\text{farmer}(x) \land \text{donkey}(x) \land \text{own}(x, y)] \rightarrow \text{beats}(x', y') \)

where the existential quantifiers have scope only over the antecedent of the conditional. This analysis leaves the pronouns unbound; clearly it does not deliver the right meaning.

Similar problems arise with indefinite antecedents in relative clauses:

(18) Every man who owns a donkey beats it.
Exercise 2. Give a representation in $L_\lambda$ capturing the intuitively correct truth conditions for (18). Then give an LF tree and a derivation for (18) using the assumptions that we have built up so far. Does this derivation give an equivalent result? If so, explain. If not, give a situation (including a particular assignment function) where one would be true but the other would be false.

According to Geach (1962), we must simply stipulate that indefinites are interpretable as universal quantifiers that can have extra-wide scope when they are in conditionals or in a relative clause. But this is more of a description of the facts than an explanation for what is happening. Moreover, it is not as if just any relative clause allows for a wide-scope universal reading of an indefinite within it:

\begin{align*}
(19) & \quad \text{A friend of mine who owns a donkey beats it.} \\
\text{There is no wide-scope universal reading for } & \text{a donkey here.}
\end{align*}

Heim (1982) idea is that indefinites have no quantificational force of their own, but are like 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:

\begin{align*}
(20) & \quad \text{In most cases, if a table has lasted for 50 years, it will last for 50 more.} \\
& \quad \iff \text{Most tables that have lasted for 50 years will last for another 50.} \\
(21) & \quad \text{Sometimes, if a cat falls from the fifth floor, it survives.} \\
& \quad \iff \text{Some cats that fall from the fifth floor survive.} \\
(22) & \quad \text{If a person falls from the fifth floor, he or she will very rarely survive.} \\
& \quad \iff \text{Very few people that fall from the fifth floor survive.}
\end{align*}
However, on Heim’s view, indefinites are unlike pronouns in that they introduce a ‘new’ referent, while pronouns pick up an ‘old’ referent. This idea of novelty is formulated in the context of dynamic semantics, where as a sentence or text unfolds, we construct a representation of the text using discourse referents. A pronoun picks out an established discourse referent. An indefinite contributes a new referent, and has no quantificational force of its own. The quantificational force arises from the indefinite’s environment.

The idea of a discourse referent is laid out by Karttunen (1976), which opens as follows:

Consider a device designed to read a text in some natural language, interpret it, and store the content in some manner, say, for the purpose of being able to answer questions about it. To accomplish this task, the machine will have to fulfill at least the following basic requirement. It has to be able to build a file that consists of records of all the individuals, that is, events, objects, etc., mentioned in the text and, for each individual, record whatever is said about it.

Karttunen characterizes discourse referents as follows: “the appearance of an indefinite noun phrase establishes a discourse referent just in case it justifies the occurrence of a coreferential pronoun or a definite noun phrase later in the text.” 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

---

3 Here, Karttunen is using “coreference” in a looser manner than the one Heim & Kratzer (1998) advocate when they say that “coreference implies reference”. For Karttunen, any kind of anaphor-antecedent relationship qualifies as coreference, even if reference does not take place.
placeholder for an individual, very much like a variable. According to Karttunen, one of the virtues of this notion is that it “allows the study of coreference to proceed independently of any general theory of extralinguistic reference” (p. 367).

Karttunen (1976) also pointed out that discourse referents have a certain lifespan; they do not license subsequent anaphora in perpetuity. Here is an example where a discourse referent dies:

(23) Bill 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 (13) and (14) 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 introduce are immortal. A good theory should account for both sides of this tension.

7.2 File change semantics

Heim’s 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 (24):

(24) a. A woman was bitten by a dog.
    b. She hit him with a paddle.
    c. It broke in half.
    d. The dog ran away.
The first sentence contains two indefinites, *a woman* and *a dog*. 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 ‘woman’, and ‘bitten by 2’, and file card 2 is associated with the property ‘dog’, and ‘bit 1’. Pictorially, we can represent the situation like this:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>woman</td>
<td>dog</td>
</tr>
<tr>
<td>1</td>
<td>bitten by 2</td>
<td>bit 1</td>
</tr>
</tbody>
</table>

After the second sentence, a third card is added to the file, and the first two cards are updated thus:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>woman</td>
<td>dog</td>
<td>paddle</td>
</tr>
<tr>
<td>1</td>
<td>bitten by 2</td>
<td>bit 1</td>
<td>used by 1 to hit 2</td>
</tr>
<tr>
<td></td>
<td>hit 2 with 3</td>
<td>was hit by 1 with 3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>paddle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>used by 1 to hit 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>broke in half</td>
</tr>
</tbody>
</table>

And so forth, so that by the end of the discourse, the file looks like this:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>woman</td>
<td>dog</td>
<td>paddle</td>
</tr>
<tr>
<td>1</td>
<td>bitten by 2</td>
<td>bit 1</td>
<td>used by 1 to hit 2</td>
</tr>
<tr>
<td></td>
<td>hit 2 with 3</td>
<td>was hit by 1 with 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ran away</td>
</tr>
<tr>
<td>3</td>
<td>paddle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>broke in half</td>
</tr>
</tbody>
</table>

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 3.** Add a sentence to (24) 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. But
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.

Some people might disagree with this identification and maintain that discourse referents 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... But... an indefinite NP [introduces] a discourse referent, not a set of discourse referents.”

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 (25) 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.

<table>
<thead>
<tr>
<th>World 1</th>
<th>World 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pug bit Joan</td>
<td>Fido bit Joan</td>
</tr>
<tr>
<td>Joan hit Pug with Paddle</td>
<td>Joan hit Fido with Paddle</td>
</tr>
<tr>
<td>Paddle broke in half</td>
<td>Paddle broke in half</td>
</tr>
<tr>
<td>Pug ran away</td>
<td>Fido ran away</td>
</tr>
</tbody>
</table>

In both worlds, it is possible to find a sequence of individuals that match the descriptions. In World 1, the sequence is (Joan, Pug, Paddle) (corresponding respectively to file cards 1, 2, and 3), and in World 2, it is (Joan, Fido, 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), etc. A file is true (a.k.a. SAT-
isfiable) in a possible world if and only if there is a sequence that satisfies it in that world.

On this view, the meaning of a sentence corresponds to an update to the file in the discourse. It is not any particular file; rather the meaning of a sentence constitutes a set of instructions for updating a given file. In other words, the meaning 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 meaning 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 (Joan, Pug, Paddle) would be in the set of world-sequence pairs making up the file represented by (25). So would the pair consisting of World 2 and the sequence (Joan, Fido, Paddle). As the meaning of a sentence in dynamic framework is something that relates an input context to an output context, the meaning would thus be a relation between two sets of world-sequence pairs.

Recall that in a static framework, the meaning 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 is true with respect to world \( w \) (or 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 meaning of a sentence corresponds to a set of world-assignment pairs, the meaning of a sentence in dynamic semantics corresponds to a relation between sets of world-assignment pairs.
7.3 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 (24) would look as follows:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman(x)</td>
<td>dog(y)</td>
<td>paddle(z)</td>
</tr>
<tr>
<td>bit(y,x)</td>
<td>hit-with(x,y,z)</td>
<td>ran-away(y)</td>
</tr>
</tbody>
</table>

Just as in file card semantics, this kind of structure is thought to be built up over the course of a discourse, and the meaning 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**. Conditions can be simple, like `woman(x)`, or complex, like `¬K` or `K ⇒ 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:
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 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 $g(x)$ chased $g(y)$.

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; $I$(owns) will correspond to a relation. So $g$ verifies farmer($x$) with respect to model $M = \langle D, I \rangle$ if and only if $g(x) \in I$(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:

(26) **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

$$M, g \models \phi$$

to denote ‘$g$ verifies condition $\phi$ in model $M$’. The rule for deciding whether a given embedded verifies a condition like farmer($x$), where a predicate applies to an argument, is defined as follows:

(27) **Verification of a predication condition**

$$M, g \models \pi(x) \text{ iff } g(x) \in I(\pi)$$

where $\pi$ is a predicate symbol 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:
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 4.** 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 explain why this is so, and explain it to them so that they say ‘Aha!’.

Another kind of atomic condition is equality:

\[
\text{(28) Verification of an equality condition} \\
M, g \models x = y \iff g(x) = g(y)
\]

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 *Pedro 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 Pedro, 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:

(29) **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 *Pedro does not own a donkey* iff:

- $f$ verifies $\text{Pedro}(x)$, and
- There is no function $g$ such that: (i) $g$ extends $f$, and (ii) $g$ verifies
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 5.** Partially specify a model $M = \langle D, I \rangle$ where Pedro 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 Pedro 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.

(30) **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:

\[
\begin{array}{c|c|c}
  x & y & \Rightarrow \\
  \text{farmer}(x) & \text{donkey}(y) & \text{owns}(x,y) & \text{beats}(x,y)
\end{array}
\]

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 \( M = (D,I) \) such that the following statements hold:

\[
(32) \quad I(\text{Pedro}) = a  \\
I(\text{farmer}) = \{a,b,c\}  \\
I(\text{donkey}) = \{d,e,f\}  \\
I(\text{owns}) = \{(a,d),(b,e),(b,f)\}  \\
I(\text{beats}) = \{(a,d),(b,e),(b,f)\}
\]

Let \( f \) be the null embedding, that 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 own relation is exactly the same as the beat 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 6.** Let $g$ be the empty embedding $\varnothing$. Using the assumptions about the model given in (32), list all of the embeddings $g$ that verify the antecedent DRS in (31). For each of those embeddings, give an embedding $h$ that verify the consequent.

**Exercise 7.** Change the model specified in (32) so that the condition in (31) is not satisfied, and name the embedding $g$ that verifies the antecedent that does not have an extension $h$ that verifies the consequent.

**Exercise 8.** 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.

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...
‘accessibility’, which constrains the ‘life-span’ of discourse referents. The interested student is encouraged to start there and work backwards from the references cited there. Note also that we have not touched on how composition works, i.e., how files or DRSs are to be constructed from an LF representation. This is a tricky area, but the interested student is encouraged to consult Muskens (1996) for an elegant approach to composition in dynamic semantics, called Compositional DRT.
8 Presupposition

8.1 Definedness conditions

In Chapter 5 we saw one example of a presuppositional expression, namely the definite determiner *the*. We translated the definite determiner using *t*-expressions, which fail to denote when nothing satisfies the description. In that case, we assumed that the denotation was a special 'undefined individual', denoted #e.1

The definite article is one of many presupposition triggers, and even if we are satisfied with our treatment of the definite article, we still need a more general way of dealing with presupposition. The determiners *both* and *neither*, for example, come with presuppositions. In a context with three candidates for a job, it would be quite odd for someone to say either of the following:

(1) a. Both candidates are qualified.
    b. Neither candidate is qualified.

If there are two candidates and both are qualified, then (1a) is clearly true and (1b) is clearly false. But if there is any number of candidates other than two, then it is hard to say whether these sentences are true or false. This suggests that *both candidates* and *neither candidate* come with a presupposition that there are exactly two candidates.

1 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’.
As the reader may have noticed, we have been assiduously avoiding plural noun phrases so far, so we will ignore both and focus on neither. We can model this presupposition by treating neither as a variant of every that is only defined when its argument is a predicate with exactly two satisfiers. Let us use $|P| = 2$ (suggesting ‘the cardinality of $P$ is 2’) as a way of writing the idea that predicate $P$ has exactly two satisfiers. This is what is presupposed. To signify that it is presupposed, we will use Beaver & Krahmer’s (2001) $\partial$ ‘partial’ operator. A formula like this:

$$\partial[|P| = 2]$$

can be read, ‘presupposing that there are exactly two $P$s’. The lexical entry for neither can be stated using the $\partial$ operator as follows:

(2) $\textit{neither} \rightarrow \lambda P \lambda Q . [\partial(|P| = 2) \land \exists x . [P(x) \land Q(x)]]$

This says that neither is basically a synonym of no, carrying an extra presupposition: that there are exactly two $P$s.

In order to be able to give translations like this, we need to augment $L_\lambda$ to handle formulas containing the $\partial$ symbol. Let us call our new language $\partial L$. In this new language, $\partial(\phi)$ will be an kind of expression of type $t$. Its value will be ‘true’ if $\phi$ is true and ‘undefined’ otherwise. To implement this, we must add the following clause to our syntax rules:

(3) **Syntax of definedness conditions**

If $\phi$ is an expression of type $t$, then $\partial(\phi)$ is an expression of type $t$.

We define the semantics of these expressions as follows:

(4) **Semantics of definedness conditions**

If $\phi$ is an expression of type $t$, then:

$$\llbracket \partial(\phi) \rrbracket_{M,g} = \begin{cases} \llbracket \alpha \rrbracket_{M,g} \text{ if } \llbracket \phi \rrbracket_{M,g} = 1 \\ \# \text{ otherwise.} \end{cases}$$

$^2|P| = 2$ is short for $\exists x \exists y [x \neq y \land P(x) \land P(y) \land \neg \exists z [z \neq x \land z \neq y \land P(z)]]$. 
The lexical entry in (2) will give us the following analysis for (1b), where $\beta$-reduced variants of the translations are given at each node:

\[
\lambda Q. [\partial(|\text{candidate}| = 2) \land \forall x. [\text{candidate}(x) \to \text{qualified}(x)]]
\]

The translation for the whole sentence (at the top of the tree) should have a defined value in a model if $|\text{candidate}| = 2$ is true in the model. If it has a defined value, then its value is equal to that of $\forall x. [\text{candidate}(x) \to \text{qualified}(x)]$.

The quantifier *every* is also sometimes argued to come with a presupposition. For example, one might hesitate to judge the following sentence as true:

\[
(6) \quad \text{Every unicorn in the world is white.}
\]

Assuming that there are no unicorns in the world, this sentence is true given the analysis of *every* that we have given so far: since there are no unicorns in the world, there are no unicorns in the world that are not white, and this is sufficient to make the sentence true according to the semantics we have given. But something feels wrong with (6), suggesting that there might be something wrong with our analysis of *every*. It seems to imply, in some sense of imply (as Strawson would say), that there are unicorns.

---

\[3\] Notice that $\beta$-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.
Presupposition

But (6) does not entail that there are unicorns; we would hesitate to judge it as true or false. We can capture this using the following kind of analysis of every:

(7) \( \langle \text{every} \rangle = \lambda P \lambda Q . [ \partial (\exists x . P(x)) \land \forall x . [P(x) \to Q(x)] ] \)

This will give rise to an undefined value for (6) in models where there are no unicorns (such as the one corresponding to reality), capturing the intuition that the sentence is neither true nor false.

**Exercise 1.** Notice that *Anna isn’t the only girl* does not presuppose that there is only one girl (and in fact asserts the opposite). Explain why this is a problem for the Fregean analysis of definite descriptions.

Based on examples like this, [Coppock & Beaver (2015)](#) argue for an analysis using the following lexical entries (where \(|P| \leq 1\) means ‘the cardinality of \(P\) is less than or equal to 1’ and is technically a shorthand for a more complicated well-formed formula in first-order logic):

- \( \langle \text{the} \rangle = \lambda P \lambda x . \partial (|P| \leq 1) \land P(x) \)
- \( \langle \text{only} \rangle = \lambda P \lambda x . P(x) \land \forall y [ y \neq x \to \neg P(y)] \)

Draw a tree for *Anna isn’t the only girl* and give a compositional derivation. Explain why the truth and definedness conditions that you derive for the sentence are satisfied only when there are multiple girls. (Note that *the only girl* will turn out to be an expression of type \(\langle e, t \rangle\) under this analysis; uses of definites in argument positions are thought to involve type-shifting operations on this view.)

**Exercise 2.** In preparation for an interview with Jimmy Carter, Stephen Colbert explained that Carter was “the only president to
have lived in public housing, *other than all of them*” (emphasis added). Assuming that this is a funny joke, explain why it is funny, using $\partial L$ to articulate the contribution of the emphasized part to the meaning.

In setting up a logic with three truth values, a number of decisions have to be made. For example, what if $\phi$ is undefined and $\psi$ is true – is $\phi \land \psi$ undefined or false? If we take undefinedness to represent ‘nonsense’, then presumably the conjunction of nonsense with anything is also nonsense. Same 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 lefthand conjunct (or disjunct) is represented by the row labels and the truth value of the righthand 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.

Now, what happens when a universal quantifier scopes over a presupposition operator? Consider the following example:

(8) Every boy loves his cat.

This would be translated:

(9) $\forall x. [\text{boy}(x) \rightarrow \text{loves}(x, \iota y[\text{cat}(y) \land \text{has}(x, y)])]$
This formula will be true in a model where every element of $D$ satisfies the formula

$$\text{boy}(x) \rightarrow \text{loves}(x, \iota y[\text{cat}(y) \wedge \text{has}(x, y)])$$

This formula will give 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 scope proposition takes on contains both 1 and #, should the truth value for the universal proposition be 1 or should it be #? Different authors have advocated different answers to this question. LaPierre (1992) and Muskens (1995) 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 (1995) 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:

$$[\forall x \phi]^{M,g} = \begin{cases} 1 & \text{if } [\phi]^{M,g[x \rightarrow k]} = 1 \text{ for all } k \in D \\ # & \text{if } [\phi]^{M,g[x \rightarrow k]} = # \text{ for some } k \in D \\ 0 & \text{otherwise} \end{cases}$$

So a universal claim is false only if the scope proposition never takes on an undefined value, and is not always true.

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 quanti-
So an existential claim is true only if the scope proposition never takes on an undefined value, and is not always false.

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 $\alpha$ or $\beta$ might denote the undefined individual? We certainly don’t want it to turn out to be true 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 $\alpha$ nor $\beta$ denotes the undefined individual, then $\alpha = \beta$ is true wrt $M$ and $g$ if $\llbracket \alpha \rrbracket^{M,g} = \llbracket \beta \rrbracket^{M,g}$, and 0 otherwise.

- If one of $\alpha$ or $\beta$ denotes the undefined individual, then $\alpha = \beta$ is false.

- If both denote the undefined individual, then $\alpha = \beta$ 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 the 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 famous sentence discussed by Russell (1905):
(10) 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 denote this predicate exists. Then the translation of (10) would be:

(11) \( \neg \text{exists}(\iota x. [\text{golden}(x) \land \text{mountain}(x)]) \)

(This of course is not the only possible treatment of this example.) The semantics of exists might then be defined as follows:

(12) \( [\text{exists}(\alpha)]^{M,g} = 1 \text{ if } [\alpha]^{M,g} \neq #_e \text{ and } 0 \text{ otherwise} \)

Assuming that \( \iota x. [\text{golden}(x) \land \text{mountain}(x)] \) denotes \( #_e \), the sentence is correctly predicted to be true under this treatment.

We are now ready to give the full semantics for \( \partial L \). 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 \( D_\sigma \) to \( D_\tau \). 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 \( (D, I) \) subject to the following constraints:

- The domain of individuals \( D \) 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 \( \tau \) is a member of \( D_\tau \).

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 \( \tau \) then
\(g(u) \in D_T\). 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** (same as before)
   
   (a) If \(\alpha\) is a non-logical constant, then \(\llbracket \alpha \rrbracket^{M,g} = I(\alpha)\).
   
   (b) If \(\alpha\) is a variable, then \(\llbracket \alpha \rrbracket^{M,g} = g(\alpha)\).

2. **Predication** (same as before)
   
   If \(\alpha\) is an expression of type \((a, b)\), and \(\beta\) is an expression of type \(a\), then \(\llbracket \alpha(\beta) \rrbracket = \llbracket \alpha \rrbracket(\llbracket \beta \rrbracket)\).

3. **Equality**
   
   If \(\alpha\) and \(\beta\) are terms, then
   
   \[
   \llbracket \alpha = \beta \rrbracket^{M,g} = \begin{cases} 
   1 & \text{if } \llbracket \alpha \rrbracket^{M,g} \neq \# \text{ and } \llbracket \beta \rrbracket^{M,g} \neq \# \text{ and } \llbracket \alpha \rrbracket^{M,g} = \llbracket \beta \rrbracket^{M,g} \\
   \# & \text{if } \llbracket \alpha \rrbracket^{M,g} = \# \text{ and } \llbracket \beta \rrbracket^{M,g} = \# \\
   0 & \text{otherwise} 
   \end{cases}
   \]

4. **Connectives**
   
   The semantics of the connectives is defined as in Table 8.1.

5. **Quantification**
   
   (a) If \(\phi\) is a formula and \(v\) is a variable of type \(a\) then
   
   \[
   \llbracket \forall v \phi \rrbracket^{M,g} = \begin{cases} 
   1 & \text{if } \llbracket \phi \rrbracket^{M,g[v \to k]} = 1 \text{ for all } k \in D \\
   \# & \text{if } \llbracket \phi \rrbracket^{M,g[v \to k]} = \# \text{ for some } k \in D \\
   0 & \text{otherwise} 
   \end{cases}
   \]

   (b) If \(\phi\) is a formula and \(v\) is a variable of type \(a\) then
   
   \[
   \llbracket \exists v \phi \rrbracket^{M,g} = \begin{cases} 
   0 & \text{if } \llbracket \phi \rrbracket^{M,g[v \to k]} = 0 \text{ for all } k \in D \\
   \# & \text{if } \llbracket \phi \rrbracket^{M,g[v \to k]} = \# \text{ for some } k \in D \\
   1 & \text{otherwise} 
   \end{cases}
   \]
6. **Lambda Abstraction** (same as before)

If $\alpha$ is an expression of type $a$ and $u$ a variable of type $b$ then
\[ \langle \lambda u. \alpha \rangle \] is that function $h$ from $D_b$ into $D_a$ such that for all objects $k$ in $D_b$, $h(k) = \langle \alpha \rangle[ u \mapsto k ]$.

This is just one example of a complete system; other design choices are of course possible.

### 8.2 Projection problem

This treatment of presupposition captures the fact, discussed in the first chapter, that presuppositions *project*. 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, as discussed. Consider the following examples:

(13) If there is a king of France, then the king of France is wise.

(14) 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. In Karttunen’s terms, *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:

(15) If France is neutral, then the king of France is wise.

(16) Either France is lucky or the king of France is wise.

Karttunen gave the following generalization:

(17) When the antecedent of the conditional (the *if*-part) entails a presupposition of the consequent (the *then*-part), the presupposition gets filtered out.

In (13), 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 (15), the antecedent is *France is staying out of the war*, 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 as follows:

(18) A presupposition of one disjunct gets filtered out when the negation of another disjunct entails it.

In (14), 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 (16), the first disjunct does not entail that there is a king of France, so the presupposition does not get filtered out.

Observe that generalizations (17) and (18) use the word *entail*. In (13) and (14), 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 *strictly entail* the presupposition. Consider the following example from Karttunen (1973):
(19) Either Geraldine is not a mormon or she has stopped wearing her holy underwear.

The second disjunct (she has stopped wearing her holy underwear) presupposes that Geraldine has holy underwear. The local context for the second disjunct is the negation of the first disjunct. The first disjunct is Geraldine is not a mormon, so the local context is Geraldine is a mormon. If we assume that all mormons have holy underwear, then the antecedent entails that Geraldine has holy underwear. So Karttunen’s generalization correctly captures the fact that (29) does not presuppose that Geraldine has holy underwear.

The system that we have introduced for dealing with presuppositions might seem to predict that presuppositions will always project, since undefinedness tends to “percolates up,” so to speak. The projection problem has been dealt with elegantly using dy

amic semantics, where the meaning of a sentence is a “context change potential”: a function that can update a discourse context. This will be discussed in the next section. However, note that Beaver & Krahmer (2001) argue that presupposition projection can in fact be handled in an empirically adequate manner in a static semantics with three truth values (true, false and undefined), using the $\partial$ operator, as we have done here.

8.3 Presupposition in Dynamic Semantics

The two generalizations about when presuppositions get filtered that were just identified (in (17) and (18)) 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 worlds that the speakers 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, the possible worlds compatible with what is presupposed. This set, which I will call the CONTEXT SET, is the set of possible worlds recognized 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 $\phi$ entails another sentence $\psi$ (written $\phi \models \psi$) if and only if whenever $\phi$ is true, $\psi$ is true. Ignoring assignment functions for the moment, and speaking of possible worlds rather than models, let us say that the proposition expressed by a sentence is the set of possible worlds (i.e., models) in which the sentence is true. Then we can say that $\phi$ entails $\psi$ if and only if the proposition expressed by $\phi$ is a subset of the proposition expressed by $\psi$: every $\phi$-world is a $\psi$-world. For example, suppose that Bart is president in $w_1$, $w_2$, and $w_3$, so the proposition expressed by 'Bart is president' is:

$$\{w_1, w_2, w_3\}$$

Assume further that in every world, Bart is a child. Thus a child is president in all of these world. But there are also other worlds where Lisa, 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

$$\{w_1, w_2, w_3\} \subseteq \{w_1, w_2, w_3, w_4, w_5\}$$

'Bart 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, [Heim (1983c)](https://example.com) 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:

\[
P = \{ w_1, w_2, w_3 \} \quad \text{[‘Bart is president’]} \\
Q = \{ w_1, w_2, w_3, w_4, w_5 \} \quad \text{[‘A child is president’]} \\
R = \{ w_2, w_3 \} \quad \text{[‘Bart has a girlfriend’]} \\
S = \{ w_1, w_4 \} \quad \text{[‘Santa’s little helper is sick’]} \\
W = \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10} \} \quad \text{[the set of all worlds]}
\]

What is the conjunction of \( P \) and \( S \), the conjunction of the proposition that Bart is president and the proposition that Santa’s little helper is sick? It is the set of worlds where both propositions are true. That is the \textit{intersection} (not the union).

\[
P \cap S \\
= \{ w_1, w_2, w_3 \} \cap \{ w_1, w_4 \} \\
= \{ w_1 \}
\]

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.

In dynamic terms, we can say that the \textit{update crashes} when the presuppositions of the sentence are not satisfied. Recall from Chapter [7](https://example.com) that in dynamic semantics, the meaning of a sentence
is a context change potential, rather than a characterization of the world. Let us write:

\[ c + \phi \]

to denote the result of updating \( c \) with the proposition expressed by \( \phi \). Ignoring assignment functions, if we take the meaning of a sentence to be a set of possible worlds, the update that a sentence makes is to narrow down the set of worlds in the context set to just those in which the proposition expressed by the sentence holds. A sentence like \( \text{John is happy} \), for example, would eliminate all worlds where John is not happy from the context set.

Exercise 3. 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 \( \text{It is raining} \)?

Suppose that we have a sentence like \( \text{John's son is bald} \), 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:

(20) **Satisfaction**
Let \( P_\phi \) be the set of worlds where the presuppositions of \( \phi \) are satisfied. A context \( c \) satisfies the presuppositions of \( \phi \) if \( X \subseteq P_\phi \).

(21) **Admittance**
A context \( c \) admits \( \phi \) if and only if \( c \) satisfies the presuppositions of \( \phi \).

Now, given a context that does not satisfy the presuppositions of a given sentence, it is easy enough to repair it so that the presuppo-
sitions are taken for granted; this process is called ACCOMMODA-
TION. But the idea is nevertheless that the update cannot proceed
until the context is such that all the presuppositions of the sen-
tence are satisfied.

A simple, non-compound sentence will have a set of BASIC
PRESUPPOSITIONS. For example, All of Homer's children are bald
presupposes that Homer has at least three 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:

(22) **Admittance conditions for non-compound sentences**

If \( \phi \) is a simple, non-compound sentence, then A context \( c \)
admits \( \phi \) if and only if \( c \) satisfies the basic presuppositions
of \( \phi \). (Karttunen [1974] 184)

---

**Exercise 4.** Assume the following:

- \( P = \{ w_1, w_2, w_3 \} \) ['Bart is president']
- \( Q = \{ w_1, w_2, w_3, w_4, w_5 \} \) ['A child is president']
- \( R = \{ w_2, w_3 \} \) ['Bart has a girlfriend']
- \( S = \{ w_1, w_4 \} \) ['Santa’s little helper 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]

Suppose that the sentence \( \phi = \) All of Homer’s children are bald
presupposes that Homer has at least three children. Suppose
that in worlds \( w_1 \ldots w_8 \), Bart, Lisa and Maggie are all children
of Homer, but in \( w_9 \) and \( w_{10} \), Maggie is actually adopted, and
therefore strictly speaking not Homer’s child. So the proposition
that Homer has at least three children (call it \( K \)) is the set of
worlds \( w_1 \ldots w_8 \):

\[ K = \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8 \} \]
K is a basic presupposition of \( \phi \); 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 \( \phi \) is a simple, non-compound sentence, it is admitted in contexts \( c \) that entail \( K \).

- Suppose that \( c = P \cap S \). Does \( c \) admit \( \phi \)? Why or why not?
- Suppose instead that \( c = W \). Does \( c \) admit \( \phi \)? Why or why not?

Now, consider (again) the contrast between the following two conditional sentences:

(23) If baldness is hereditary, then John’s son is bald.
    \( \gg \) John has a son.

(24) 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:
Admittance conditions for a conditional sentence

Context $c$ admits “If $\phi$ then $\psi$” just in case (i) $c$ admits $\phi$, and (ii) $c + \phi$ admits $\psi$.

Here $c + \phi$ designates ‘$c$ updated with $\phi$’. The result of this update will be the same as if $\phi$ 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 $\phi$ is not true.

Consider the following examples:

(26) If Homer has at least 3 children, then all of his children are bald.

(27) If Homer is bald, then all of his children are bald.

Assume the following:

$A = \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8 \}$  ['Homer has $\geq 3$ 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 (26)? According to the admittance conditions for conditional sentences in (25), it does just in case (i) $c$ admits Homer has at least three children and (ii) $c + \text{Homer has at least three children}$ admits all of his children are bald. Since Homer has at least three children carries no presuppositions, the first condition is satisfied. What about the second condition? The result of updating $c$ with Homer has at least three children is the set $A$, the set of worlds where Homer indeed has at least three children. Now the question is whether this set, $A$, admits the non-compound sentence all of his children are bald. Since it is a non-compound sentence, the rule for non-compound sentences (22) applies. What all of his children are bald presupposes is that Homer has at least three children. This is satisfied in all of the worlds in $A$, so the second condition is satisfied as well. Hence $c$ does admit (26). But the same does not hold for (27).
Figure 8.1: Example propositions

**Exercise 5.** Explain step-by-step why \( c \) (as defined in the foregoing discussion) does not admit (27).

**Exercise 6.** Refer to Figure 8.1. 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 Homer has at least three children *projects* in a case like (27), and not in a case like (26). In order for the conditional as a whole to be admitted by a given context, both of the conditions in (25)
must be met. The first condition will be met only if the presuppositions of the antecedent are already satisfied 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 the situation in a case like (26), 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 $\rightarrow c+$ the antecedent
- the second conjunct in a conjunction $\rightarrow c+$ the negation of the first conjunct
- the second disjunct in a disjunction $\rightarrow c+$ the negation of the first disjunct
- the complement of a propositional attitude verb $\rightarrow$ 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:

(28) 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 (29) above, repeated here:
(29) Either Geraldine is not a mormon or she has stopped wearing her holy underwear.

Here we have a disjunction. The local context for the second disjunct is $c+$ the negation of the first disjunct. The first disjunct ($Geraldine is not a mormon$) 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 $c+$ $Geraldine is a mormon$. Suppose it is common ground in the global context that all mormons have holy underwear. Then the local context entails that Geraldine has holy underwear. The consequent, $she has stopped wearing her holy underwear$, presupposes that Geraldine has holy underwear. Since the local context entails this proposition, the global context need not entail it, so the presupposition is filtered out.

**Exercise 7.** 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 Coordination and Plurals

9.1 Coordination

Let us now consider coordination in more detail. We may include sentences with *and* and *or* among the well-formed expressions of our language by extending our syntax and lexicon as follows:

(1) Syntax

| S  | →  | S JP |
| JP | →  | J S |

(2) Lexicon

J: *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. \( \text{and}_S \sim \lambda q \lambda p. p \land q \)

b. \( \text{or}_S \sim \lambda q \lambda p. p \lor q \)

This will work for coordinations of sentences. For example, here is a tree for *Homer smokes and Marge drinks*:
Exercise 1. Perform the beta reductions involved in Tree 4. This exercise can be solved in the Lambda Calculator.

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. John 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.

An early style of analysis consisted in analyzing all coordinations as underlyingly sentential, even those of constituents other than sentences. For example, VP coordination was analyzed as involving deletion of the subject of the second sentence is silent (indicated here as strikethrough):

(6)  

a. Homer smokes and drinks.  

b. Homer smokes and Homer 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.  

\[ \exists x. \text{smokes}(x) \land \text{drinks}(x) \]

b. Somebody smokes and somebody drinks.  

\[ \exists x. \text{smokes}(x) \land \exists x. \text{drinks}(x) \]

(8)  

a. Everybody smokes or drinks.  

\[ \forall x. \text{smokes}(x) \lor \text{drinks}(x) \]

b. Everybody smokes or everybody drinks.  

\[ \forall x. \text{smokes}(x) \lor \forall x. \text{drinks}(x) \]

**Exercise 2.** 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:
Coordination and Plurals

(9) **Syntax**

\[
\begin{align*}
X & \rightarrow X \text{JP} \\
\text{JP} & \rightarrow J X \quad \text{where } X \in \{\text{S, VP, DP, V, ...}\}
\end{align*}
\]

The semantic side is trickier. It is not obvious if we can give a single meaning 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.  \( \text{and}_{VP} \sim \lambda P' \lambda P \lambda x. P(x) \land P'(x) \) 

b.  \( \text{or}_{VP} \sim \lambda P' \lambda P \lambda x. P(x) \lor P'(x) \) 

This tree shows the entry for *and* in action. The result is what we want: the quantifier *somebody* takes scope over *and*. 

The following entries can be used to coordinate transitive verbs.

\[(12) \quad \text{a. } \text{and}_{V} \sim \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \land R'(y)(x) \]
\[\quad \text{b. } \text{or}_{V} \sim \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \lor R'(y)(x) \]

Exercise 3. Using the lexical entry above, draw the tree for John caught and ate the fish. This exercise can be solved in the Lambda Calculator.

We can approach noun phrase coordination in this way as well. Let us first look at conjunctions of quantifiers:
Coordination and Plurals

(13) a. Every man and every woman arrived.
   \[ ∀x.\text{man}(x) \to \text{arrived}(x) \land ∀x.\text{woman}(x) \to \text{arrived}(x) \]

   b. A man or a woman arrived.
   \[ ∃x.\text{man}(x) \land \text{arrived}(x) \lor ∃x.\text{woman}(x) \land \text{arrived}(x) \]

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. \[ \text{and}_{DP} \sim \lambda Q' ΛQΛP . Q(P) \land Q'(P) \]

   b. \[ \text{or}_{DP} \sim \lambda Q' ΛQΛP . Q(P) \lor Q'(P) \]

**Exercise 4.** Using the lexical entries above, draw the trees for *Every man and every woman arrived* and *A man or a woman arrived*. This exercise can be solved in the Lambda Calculator.

In all of the examples so far, we have always coordinated two constituents 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-*⟨(e, t), t⟩* noun phrase.

(15) John and every woman arrived.
   \[ \text{arrived}(j) \land [∀x.\text{woman}(x) \to \text{arrived}(x)] \]

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 *⟨(e, t), t⟩* for every translation of type *e*:
Type-Shifting Rule 5. Entity-to-quantifier shift
If $\alpha \sim \langle \alpha \rangle$, where $\langle \alpha \rangle$ is of category $e$, then:

$$\alpha \sim \lambda P. P(\langle \alpha \rangle)$$

as well.

This rule, which goes back to [Montague (1973)], 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 $John \sim j$ then also $John \sim \lambda P. P(j)$. 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 5. Draw the tree for $John and every woman arrived$. This exercise can be solved in the Lambda Calculator.

So far, we have been able to keep the number of entries for each conjunction down to one per syntactic category. In fact, we can do better than that. All of the entries for conjunction and for disjunction have $\land$ and $\lor$ at their core respectively. And all of them operate on types that end in $t$, namely $\langle e, t \rangle$ for VP coordination, $\langle e, \langle e, t \rangle \rangle$ for coordination of transitive verbs, and $\langle \langle e, t \rangle, t \rangle$ for DP coordination. The following recursive definitions will work for every type that ends in $t$. For more details on this approach, see for example [Partee & Rooth (1983b)] and [Winter (2001b)].

$$\langle \langle e, \langle e, t \rangle \rangle, \langle e, \langle e, t \rangle \rangle \rangle = \lambda X_T \lambda Y_T \lambda Z_{\sigma_1} \langle \langle \langle e, \langle e, t \rangle \rangle, \langle e, \langle e, t \rangle \rangle \rangle \rangle (X(Z))(Y(Z)) \quad \text{if } \tau = \langle \sigma_1, \sigma_2 \rangle$$

$$\langle \langle e, \langle e, t \rangle \rangle, \langle e, \langle e, t \rangle \rangle \rangle = \lambda q \langle q, q \rangle \quad \text{if } \tau = \langle e, e, t \rangle$$

$$\langle \langle e, \langle e, t \rangle \rangle, \langle e, \langle e, t \rangle \rangle \rangle = \lambda p \langle p, p \rangle \quad \text{if } \tau = \langle e, e, e \rangle$$
For example, here is how the schema in (16) derives DP-coordinating and. The type of DP (after lifting entities to quantifiers if necessary) is $\langle \langle e, t \rangle, t \rangle$. So the type of DP-coordinating and is $\langle \tau, \langle \tau, \tau \rangle \rangle$, where $\tau = \langle \langle e, t \rangle, t \rangle$. Since $\tau \neq t$, we look for $\sigma_1$ and $\sigma_2$ such that $\tau = \langle \sigma_1, \sigma_2 \rangle$. This works for $\sigma_1 = \langle e, t \rangle$ and $\sigma_2 = t$. We plug in these definitions into the last line of (16) and get:

\[
\lambda X_{\langle \langle e, t \rangle, t \rangle} \lambda Y_{\langle \langle e, t \rangle, t \rangle} \lambda Z_{\langle e, t \rangle} \cdot \langle \text{and} \rangle_{\langle t, \langle t, t \rangle \rangle} (X(Z))(Y(Z))
\]

To resolve $\langle \text{and} \rangle_{\langle t, \langle t, t \rangle \rangle}$, we apply Definition (16) once more. This time, $\tau = t$, so the result is simply logical conjunction:

\[
\text{and}_{\langle t, \langle t, t \rangle \rangle} \sim \lambda q \lambda p \cdot p \land q
\]

We plug this into the previous line and get the final result:

\[
\lambda X_{\langle \langle e, t \rangle, t \rangle} \lambda Y_{\langle \langle e, t \rangle, t \rangle} \lambda Z_{\langle e, t \rangle} \cdot Y(Z) \land X(Z)
\]

This is indeed equivalent our entry for DP-coordinating and in (14a). The entry will only work if both DPs are of type $\langle \langle e, t \rangle, t \rangle$. If necessary, one or both DPs may need to be lifted into that type first by applying the type shifter above.

\[
\text{Exercise 6. Draw the tree for } \textit{Homer and Marge smoke} \text{ and make sure you get the result above. You will need to apply the type shifter once on each conjunct. This exercise can be solved in the Lambda Calculator.}
\]
9.2 Mereology

All of the occurrences of *and* that we have seen so far can be related to the meaning of logical conjunction. The schema in [16] encapsulates this relation by reducing various uses of *and* to logical conjunction. This will not work in every case, though.

(22)  

a. Homer and Marge are a happy couple.

b. Homer and Marge met (in the park).

There is no obvious way to formulate the truth conditions of (22a) or (22b) using logical conjunction. For example, (22b) cannot be represented as \( \text{met}(h) \land \text{met}(m) \), since this would entail \( \text{met}(h) \) as well as \( \text{met}(m) \). In other words, it would have the nonsensical entailments that Homer met and that Marge met. This is nonsensical because a singular individual can't meet. Only two or more people can. Similarly, only two people can be a happy couple. Predicates like *meet* and *be a couple* are called collective. They apply to collections of individuals directly, without applying to those individuals. In the sentences above, 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” Homer-and-Marge.

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 *Homer and Marge* would be represented as the set that contains just these two individuals. Instead, we will extend our formal toolbox by borrowing from mereology, the study of parthood. There are many reasons for this choice. One is that using mereology for this purpose has been standard practice in formal semantics since Link (1983). Another reason is that set theory makes formal distinctions that turn out not to be needed in mereology. Where set theory is founded on two relations (\( \in \) and \( \subseteq \)), mereology collapses them into one, the parthood
relation. This relation holds both between Homer and Homer-and-Marge (where in set theory, we would use $\epsilon$), and also between Homer-and-Marge and Homer-and-Marge-and-Bart (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 Homer-and-Marge is represented formally as the sum $h \oplus m$. Collective predicates apply directly to such sums. For example, *Homer and Marge met* can be represented as $\text{met}(h \oplus m)$. Since the sum $h \oplus m$ is of type $e$, the type of the VP is $(e, t)$ as usual.

**Exercise 7.** Formulate an additional lexical entry for $\text{and}_{DP}$ that conjoins two entities of type $e$ and returns their sum. Draw the tree for *Homer and Marge met*. This exercise can be solved in the Lambda Calculator.

Mereology can be axiomatized in a way that gives rise to algebraic structures. An algebraic structure is essentially a set with a binary operation (in this case, $\oplus$) defined on it. Figure 9.2 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 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 9.2, \( t \) is part of \( t \oplus d \), and \( t \oplus d \) is part of \( t \oplus d \oplus h \); therefore, by transitivity, \( t \) is also part of \( t \oplus d \oplus h \). 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 9.1: 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 figures like (9.2), the sum of any nonempty set of individuals \( P \) is always the lowest individual that sits above every element of \( P \). (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.

### 9.3 The plural

Coordinations of proper nouns are not the only way to talk about sums:

(23)  
\begin{align*}
  &\text{a. Tom, Dick and Harry met.} \\
  &\text{b. Some boys met.} \\
  &\text{c. Three boys met.} \\
  &\text{d. The boys met.}
\end{align*}

In each of these three sentences, the collective predicate \textit{met} applies to a sum $x$. Only (23a) fully specifies the parts of that sum, while (23b) and (23d) described it partially. What is the meaning of the noun \textit{boys}? One way to describe it is in terms of the conditions it imposes on $x$, namely, \textit{boys} requires it to be the sum of a set of boys. In general, the meaning of a plural noun can be described in terms of the meaning 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 meaning of the plural (Link, 1983):

\begin{definition}
The algebraic closure $\ast P$ of a set $P$ is the set that contains any sum of any nonempty subset of $P$.
\end{definition}

The most straightforward way to implement this idea is to identify the meaning of the plural morpheme with the “star operator”:
(25) \(-s \sim \lambda P. \ast P\)

For example, suppose that we are in a model with just three boys, Tom, Dick and Harry. Then the denotation of the noun \textit{boy} might be modeled as \(\{t, d, h\}\). The denotation of the noun \textit{boys} is the algebraic closure of that set: \(\{t, d, h, t \oplus d, t \oplus h, d \oplus h, t \oplus d \oplus h\}\). 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 \textit{Tom are boys}, and the sentence \textit{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 \textit{boys} essentially synonymous with \textit{two or more boys}. But \textit{No doctors are in the room} is not synonymous with \textit{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 (25) as the meaning of the plural, and rule out \textit{Tom are boys} on pragmatic grounds. That is, \textit{boys} literally means \textit{one or more boys}, and the singular is a special case of the plural. \textit{Boy} and \textit{boys} are in competition, and the singular form blocks the plural form because it is more specific \cite{Sauerland2005, Spector2007}.

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 \textit{meet} as predicates that apply directly to sum individuals:

(26) \begin{align*}
&\text{a. Tom (and) Dick and Harry met. } \sim \text{ meet}(t \oplus d \oplus h) \\
&\text{b. Some boys met. } \sim \exists x. \ast \text{boy}(x) \land \text{meet}(x)
\end{align*}

As seen in (26a), 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:

(27) \begin{align*}
&\text{a. Tom is a boy and Dick is a boy. } \sim \text{boy}(t) \land \text{boy}(d) \\
&\text{b. Tom and Dick are boys. } \sim \ast \text{boy}(t \oplus d)
\end{align*}
Exercise 8. Draw trees for the sentences in (26) and (27), using the appropriate entries for and in each case. You can use the same entry for some as in Chapter (5). Assume that is, a and are denote identity functions, or treat them as vacuous nodes. Make sure that the result is as in (26) and (27). This exercise can be solved in the Lambda Calculator.

If binary sum is realized as and, is there a linguistic expression that is able to build the sum of more than two individuals? The definite article the, when used with a plural noun as in sentence (23d), comes to mind as a possible candidate. That is, the boys could correspond to the sum of all the entities to which the predicate denoted by boys applies. In a model where the boys are Tom, Dick and Harry, this is equivalent to \{t, d, h, t\oplus d, t\oplus h, d\oplus h, t\oplus d\oplus h\}. As can be checked with Figure (9.2) the sum of this set, and therefore the denotation of the boys, is just t\oplus d\oplus h. This is exactly what we want.

But in other cases, such as the boy and the two boys, we run into a problem. 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:

\[(28) \quad \text{two} \sim \lambda x. \text{card}(x) = 2\]

In order to deal with sentences like Two boys met as well as The two boys met, we will assume that in the first case there is a silent determiner with the semantics of a generalized existential quantifier:

\[(29) \quad \emptyset_D \sim \lambda P \lambda P'. \exists x. [P(x) \land P'(x)]\]

In our model, the set denoted by two boys is \{t \oplus d, t \oplus h, d \oplus h\}. As a glance at Figure (9.2) will confirm, the sum of this set is
t ⊕ d ⊕ h. So we end up with the rather odd prediction that *the two boys* refers to this sum!

Intuitively, *the two boys* should be a presupposition failure, because there are three boys in our model. So we should build a source of presupposition failure into our meaning 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]:

\[
(30) \quad \text{the } ≈ \lambda \mu \nu \nu [P(\nu) \land \forall x[P(x) \rightarrow x \leq \nu]]
\]

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 (30) picks out the only member of that singleton. In all other models, the *ι* operator will not be defined.

**Exercise 9.** In the model where the boys are Tom, Dick, and Harry, what (if anything) do the expressions *the boy*, *the boys*, *the two boys* and *the three boys* denote? In each case, explain which presupposition arises and whether it is satisfied.

**Exercise 10.** Translate *The boys met, Two boys met, and The two boys met.* This exercise can be solved in the Lambda Calculator.

We have now seen two ways to treat conjunction of proper names. We can either represent it as ⊕ and combine the two individuals directly, or we can lift the individuals to generalized quantifiers and use the schema in [16]. The two interpretive schemes lead to different representations:

\[
(31) \quad \text{a. John and Bill arrived. } ≈ \text{arrive}(j) \land \text{arrive}(b) \]
\[
\text{b. John and Bill arrived. } ≈ \text{arrive}(j \oplus b)
\]
What (31b) is that arrive holds of the collective individual \( j \oplus b \). To make sense of this, we can assume that a collective individual arrives if and only if all the atomic individuals arrive that it consists of. Predicates with this property are called distributive. As an alternative, we can assume that arrive only holds of atomic individuals, but that whenever it combines with a plural noun phrase, it first combines with a silent plural morpheme \( \varnothing \) which has the same meaning as the plural morpheme [Landman, 1989]:

(32) \( -\varnothing \sim \lambda P. \ast P \)

On this approach, the relationship between sentential conjunction and conjunction of individuals can be modeled in a parallel way no matter if they combine with a noun (as in (27) with is a boy / are boys) or with a verb (as in the following pair):

(33) a. John arrived and Bill arrived. \( \sim \) arrive\((j) \land \) arrive\((b) \)  
b. John and Bill arrived-\( \varnothing \). \( \sim \ast \) arrive\((j \oplus b) \)

The idea is that if arrive applies to each of a given set of individuals, say a set that contains among others \( j \) and \( b \), then \( \ast \) arrive applies to any sum drawn from that set, say \( j \oplus b \). In fact, some researchers assume that the star operator is inserted on every verb even when it combines with a singular subject [Križka, 1998; Kratzer, 2007]:

(34) John arrived. \( \sim \ast \) arrive\((j) \)

This might seem surprising but it works because \( \ast \) arrive will be true of anything arrive is also true of. Once again, the singular is a special case of the plural.

9.4 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 \textit{arrived-Ø}. All these are one-place predicates. Sums can also be related by two-place predicates, as in the following sentences:

(35) a. The men in the room are married to the girls 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 gadgets.

Let us take a closer look at the ways the plural entities in these sentences are related. Sentence (35a) is true in a scenario where each of the men in the room is married to one of the girls across the hall, and each of the girls is married to one of the men. Sentence (35b) (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 (35c) is true in a scenario where Tom, Dick and Harry own gadgets in such a way that a total of four gadgets are owned. A widespread view is that these scenarios corresponds to genuine readings of these sentences, rather than special circumstances under which they are true. These readings are then called \textit{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 gadgets. 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):
(36) **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.

(37) **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 $R(x_1)(x_2)\ldots(x_n)$.

(38) **Definition: Algebraic closure of an n-place predicate**
The algebraic closure $^\ast R$ of an $n$-place predicate $R$ is the set that contains any sum of any nonempty subset of tuples of $R$. We write $^\ast R(a, b)$ for $^\ast R(\langle a, b, \rangle)$.

We can then represent cumulative readings by using the algebraic closure of transitive verbs:

(39) Tom, Dick and Harry own four gadgets. $\rightarrow$

$$\exists x_1. \, ^\ast \text{gadget}(x) \land \text{card}(x) = 4 \land
^\ast \text{own}(t \oplus d \oplus h, x)$$

An example model which verifies formula 38 is the one described above, where Tom owns gadget 1, Dick owns gadget 2, and Harry owns gadgets 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 $\langle t \oplus d \oplus h, g_1 \oplus g_2 \oplus g_3 \oplus g_4 \rangle$.

### 9.5 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 figures like [9.2] 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 (27b) logically follows from (27a), 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 semantics 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. There are different formulations of CEM. Here is one, based on 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.

(40) **Axiom: Transitivity**
Any part of any part of a thing is also part of that thing.

(41) **Definition: Proper parthood**
If something is part of a thing but not identical to it, we say that it is a proper part of that thing.

(42) **Definition: Disjointness**
If two things do not have any part in common, we say that they are disjoint.

(43) **Axiom: Weak supplementation**
Any proper part of any thing is disjoint from one of the parts of that thing.

(44) **Definition: Upper bound**
Something of which everything in P is part is called an upper bound of P.

(45) **Definition: Least upper bound / Sum**
If an upper bound of P is part of every upper bound of P, we call it a least upper bound of P, or a sum of P.
Coordination and Plurals

Axiom: Existence of sums
Every nonempty P has a sum.

Axiom: Filtration
Every part of any sum of P has a part in common with something in P.

These axioms jointly define classical extensional mereology. Some theorems we can prove from these axioms:

Theorem: Reflexivity
Everything is part of itself. (TODO: prove this.)

Theorem: Antisymmetry
Two distinct things cannot both be part of each other. (Proof: Suppose otherwise. Then there are two distinct things \(a\) and \(b\) that are both part of each other. Since they are distinct, they must be proper parts of each other. By Axiom (43), there is a part of \(b\) from which \(a\) is disjoint. Call that part \(c\). Since \(b\) is part of \(a\), and \(c\) is part of \(b\), by Axiom (40) \(c\) is part of \(a\). By Theorem (48), \(a\) is also part of \(a\). So \(a\) and \(c\) have a part in common, namely \(a\), and cannot be disjoint. Contradiction.)

Theorem: Uniqueness of sums
Every nonempty set has exactly one sum. (Proof: Suppose otherwise. Then there is a nonempty set \(S\) that has two distinct sums \(a\) and \(b\). By definition (45), each of these sums is an upper bound of \(S\) and is part of every upper bound of \(S\). So each of these sums is part of the other one. This contradicts Theorem (49).)

Theorem: Associativity of sums
The sum of the sum of two things with a third thing is the same as the sum of one of these things with the sum of the second of these things with the third thing. In other words, the order in which you sum up three things doesn't matter. (TODO: prove this.)

Theorem (50) justifies our decision to talk about “the sum” rather than “a sum” of a set P. Now we can say more precisely what
we mean with the notation we have already introduced informally above: We write $\oplus P$ for that sum. In the special case where $P$ consists of just two members, $a$ and $b$, we abbreviate $\oplus \{a, b\}$ as $a \oplus b$.

We can now prove 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 $\text{boy}(j) \land \text{boy}(b)$ entails the conclusion $^{*}\text{boy}(j \oplus b)$. According to Definition [24], $^{*}\text{boy}$ is the set that contains any sum of any nonempty set of boys. So, $^{*}\text{boy}(j \oplus b)$ is true if and only if $j \oplus b$ is the sum of some nonempty set of boys. The obvious candidate is $\{j, b\}$. So we need to show two things: that $\{j, b\}$ is a nonempty set of boys, and that $j \oplus b$ is its sum. By assumption, $\text{boy}(j) \land \text{boy}(b)$, hence $j$ and $b$ are boys. So $\{j, b\}$ is a nonempty set of boys. That $j \oplus b$ is the sum of this set follows from the definition of $\oplus$. This concludes the proof.

### 9.6 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 ($L_{\lambda}$ with three truth values as in Chapter 8), plus the following additions:

#### 9.6.1 Logic syntax

We add the following primitive symbol to our syntax.

1. **Parthood** If $\alpha$ and $\beta$ 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 $\phi$ is an expression of type $(e, t)$, we write $\oplus \phi$ (read as: “the sum of $\phi$”) for the expression $[\forall x.[\forall y.[\phi(y) \rightarrow y \leq x]] \land [\forall z.[\forall z'.[\phi(z') \rightarrow z' \leq z]] \rightarrow x \leq z]]$. (That is, $\oplus \phi$ denotes
the least upper bound, or sum, of \( \phi \) – see Definition (45).
By Axiom (50), this will be defined whenever \( \phi \) applies to at least one entity.)

2. If \( \alpha \) and \( \beta \) are terms of type \( e \), we write \([\alpha \oplus \beta]\) (read as: “the sum of \( \alpha \) and \( \beta \)” for the expression \( \bigoplus[\lambda x. x = \alpha \lor x = \beta] \).

3. An expression of the form \([\alpha \oplus \beta] \oplus \gamma\) or \([\alpha \oplus [\beta \oplus \gamma]]\) can be simplified to \([\alpha \oplus \beta \oplus \gamma]\).

9.6.2 Logic semantics

Expressions are interpreted with respect to both:

- a model \( M = \langle D, I, \preceq \rangle \) where \( D \) and \( I \) are defined as usual and \( \preceq \) is the parthood relation over individuals that obeys the conditions listed above,
- an assignment \( g \) defined as usual.

For every well-formed expression \( \alpha \), \( \llbracket \alpha \rrbracket_{M,g} \), is defined recursively as usual. We add the following rule:

(52) Parthood
If \( \alpha \) and \( \beta \) are expressions of type \( e \), then \( \llbracket \alpha \leq \beta \rrbracket = 1 \) if \( \llbracket \alpha \rrbracket \leq \llbracket \beta \rrbracket \), otherwise \( \llbracket \alpha \leq \beta \rrbracket = 0 \).

9.6.3 English syntax

Syntax rules. We add the following rules for coordination:

(53) Syntax
\[
\begin{align*}
X & \rightarrow \ X \ JP \\
JP & \rightarrow \ J \ X \\
\end{align*}
\]
where \( X \in \{ S, VP, DP, V, \ldots \} \)

In addition, we add the following rule for the plural:

\[
\begin{align*}
N & \rightarrow \ N \ Pl
\end{align*}
\]
Lexicon. Lexical items are associated with syntactic categories as follows:

- **D:** $\emptyset_D$
- **A:** *two, three* etc.
- **J:** *and, or*
- **Pl:** -s
- **V:** *met, own*

### 9.6.4 Translations

**Type $\langle e, t \rangle$:**

1. $\text{smokes} \leadsto \lambda x. \cdot \text{smoke}(x)$
2. $\text{drinks} \leadsto \lambda x. \cdot \text{drink}(x)$
3. $\text{two} \leadsto \lambda x. \cdot \text{card}(x) = 2$

**Type $\langle e, \langle e, t \rangle \rangle$:**

1. $\text{caught} \leadsto \lambda y \lambda x. \cdot \text{catch}(x, y)$
2. $\text{ate} \leadsto \lambda y \lambda x. \cdot \text{eat}(x, y)$
3. $\text{own} \leadsto \lambda y \lambda x. \cdot \text{own}(x, y)$

**Type $e$:**

1. $\text{Homer} \leadsto h$
2. $\text{Marge} \leadsto m$
3. $\text{Tom} \leadsto t$
4. $\text{Dick} \leadsto d$
5. $\text{Harry} \leadsto h$

**Type $\langle t, \langle t, t \rangle \rangle$:**
1. $and_S \sim \lambda q \lambda p. p \land q$

2. $or_S \sim \lambda q \lambda p. p \lor q$

Type $\langle (e, t), (e, t) \rangle$:

1. $is \sim \lambda P. P$

2. $a \sim \lambda P. P$

3. $and_{VP} \sim \lambda P' \lambda P \lambda x. P(x) \land P'(x)$

4. $or_{VP} \sim \lambda P' \lambda P \lambda x. P(x) \lor P'(x)$

Type $\langle (e, \{e, t\}), (e, \{e, t\}) \rangle$:

1. $and_V \sim \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \land R'(y)(x)$

2. $or_V \sim \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \lor R'(y)(x)$

Type $\langle (e, t), e \rangle$:

1. $the \sim \lambda P \pi z[P(z) \land \forall x[P(x) \rightarrow x \leq z]]$

Type $\langle (e, t), (e, t) \rangle$:

1. $-s \sim \lambda P. \neg P$

Type $\langle (e, t), \langle (e, t), t \rangle \rangle$:

1. $\exists D \sim \lambda P \lambda Q. \exists x.[P(x) \land Q(x)]$
10 | Event semantics

10.1 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. Homer smokes and Marge drinks.
    b. $\Rightarrow$ Homer smokes.

This argument is captured by the following logical entailment:

(2) a. $\text{smokes}(h) \land \text{drinks}(m)$
    b. $\text{smokes}(h)$

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. $\Rightarrow$ Jones buttered the toast.

Here is how we would represent (3a) given the previous chapters (we are treating the toast as if it was a constant rather than a definite description, but nothing will hinge on this):
If this representation is correct, (3a) is about only two entities: Jones and the toast. Which entity does *slowly* describe in (3b)? Is it perhaps Jones who is slow? Then we might represent the meaning of that sentence as follows:

\[(5) \quad \text{butter}(j, t) \land \text{slow}(j)\]

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.

\[
\begin{align*}
\text{(6)} & \quad \text{a. Jones buttered the bagel quickly.} \\
& \quad \text{b. butter}(j, b) \land \text{quick}(j)
\end{align*}
\]

But then, in any model where (5) and (6b) are both true, the following will also be true as a matter of logical consequence!

\[(7) \quad \text{slow}(j) \land \text{quick}(j)\]

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:

\[
\begin{align*}
\text{(8)} & \quad \text{a. Jones’ buttering of the toast was artful.} \\
& \quad \text{b. It happened slowly.}
\end{align*}
\]
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), their correct logical representations are not (5) and (6b) but rather something like the following, where $e$ ranges over events:

(9)  
a. $\exists e. \text{butter}(j, t, e) \land \text{slow}(e)$  
b. $\exists e. \text{butter}(j, b, e) \land \text{quick}(e)$

A sentence like (3a) would then be represented as:

(10) $\exists e. \text{butter}(j, t, e)$

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. $\Rightarrow$ Jones buttered the toast slowly in the kitchen.  
c. $\Rightarrow$ Jones buttered the toast slowly.  
d. $\Rightarrow$ Jones buttered the toast.
Event semantics provides a straightforward account of these entailment patterns:

\[(12)\]  
\[a. \exists e. \text{butter}(j, t, e) \land \text{slow}(e) \land \text{loc}(e, k) \land \text{time}(e, \text{noon})\]  
\[b. \exists e. \text{butter}(j, t, e) \land \text{slow}(e) \land \text{loc}(e, k)\]  
\[c. \exists e. \text{butter}(j, t, e) \land \text{slow}(e)\]  
\[d. \exists e. \text{butter}(j, t, e)\]

**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]:

\[(13)\]  
\[a. \text{John saw Mary leave.}\]  
\[b. \Rightarrow \text{Mary left.}\]  

\[(14)\]  
\[a. \exists e \exists e'. \text{saw}(j, e, e') \land \text{leave}(m, e')\]  
\[b. \exists e'. \text{leave}(m, e')\]

**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]:

\[(15)\]  
\[a. \text{Brutus stabbed Caesar violently.}\]  
\[b. \Rightarrow \text{There was something violent.}\]  

\[(16)\]  
\[a. \exists e. \text{stab}(b, c, e) \land \text{violent}(e)\]  
\[b. \exists e. \text{violent}(e)\]

### 10.1.1 The Neo-Davidsonian turn

As we have seen, Davidson equipped verbs with an additional event argument. Later authors, however, take 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 small 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. A commonly held view on thematic roles is that they encapsulate generalizations over shared entailments of argument positions in different predicates. For example, the agent initiates the event, or is responsible for the event; the theme undergoes the event; the instrument is used to perform an event; the beneficiary is the entity for which the event was performed; and so on. Additional thematic roles that specify the location of an event in space and time are often proposed. 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).

On the Neo-Davidsonian view, Jones buttered the toast might be represented as follows:

\[ (17) \exists e. butter(e) \land agent(e, j) \land theme(e, t) \]

In Neo-Davidsonian event semantics, there is no fundamental semantic distinction between syntactic arguments such as the subject and object of a verb, and syntactic adjuncts such as adverbs and prepositional phrases. For example, in the following representation of Jones buttered the toast with a knife, the conjunct that represents the prepositional phrase is essentially parallel to those conjuncts that represent Jones and the toast. (For simplicity, we represent a knife as if it was a constant. As in the case of the toast, this is not essential.)

\[ (18) \exists e. butter(e) \land agent(e, j) \land theme(e, t) \land with(e, k) \]

One of the advantages of the Neo-Davidsonian view is that it allows us to capture semantic entailment relations between dif-
different syntactic subcategorization frames of the same verb, such as causatives and their intransitive counterparts [Parsons 1990]:

(19) a. Mary felled the tree.
    b. ⇒ The tree fell.

(20) a. \( \exists e. \text{fall}(e) \land \text{agent}(e, m) \land \text{theme}(e, t) \)
    b. \( \exists e. \text{fall}(e) \land \text{theme}(e, t) \)

(21) a. Mary opened the door.
    b. ⇒ The door opened.

(22) a. \( \exists e. \text{open}(e) \land \text{agent}(e, m) \land \text{theme}(e, d) \)
    b. \( \exists e. \text{open}(e) \land \text{theme}(e, d) \)

The Neo-Davidsonian approach raises important questions, many of which are 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 (23a) – perhaps beneficiary – matches the one of Mary in (23b).

(23) a. John gave the ball to Mary.
    b. John gave Mary the ball.

We might think of this role as the meaning of to in (23a), but in (23b) 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 silent functional heads, often called *theta roles* [Chomsky 1995].

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 (24).

\[ \exists e. \text{butter}(e) \land \text{agent}(e) = j \land \text{theme}(e) = t \]

A differing view holds that one can touch a man and his shoulder in the same event (Kriška 1992).

### 10.2 Quantification in 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 be-
tween events and their internal arguments while external arguments (subjects) are related to verbs indirectly by theta roles.

10.2.1 The first strategy: Verbs as events

On the first strategy, verbs denote predicates of events:

(25)  a.  \( \text{bark} \sim \lambda e. \text{bark}(e) \)
    b.  \( \text{butter} \sim \lambda e. \text{butter}(e) \)
    c.  \ldots

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 \( \theta \).

(26)  Syntax
  \[
  \text{DP} \rightarrow \theta \text{DP}
  \]

We then write lexical entries that map these heads suitable roles:

(27)  Lexicon
  \( \theta \): [agent], [theme], \ldots

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 \( \theta \) head gets mapped to the “right” role.

Next, we map these theta roles to thematic roles:
(28)  a.  \[\text{[agent]} \sim \lambda x \lambda e. \text{agent}(e) = x\]
    b.  \[\text{[theme]} \sim \lambda x \lambda e. \text{theme}(e) = x\]
    c.  \[\ldots\]

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 \((v, t)\) is the type of an event predicate.

**Type-Shifting Rule 6. Existential closure**

If \(\alpha \sim \llangle \alpha \rrangle\), where \(\llangle \alpha \rrangle\) is of category \((v, t)\), then:

\[
\alpha \sim \exists e. \llangle \alpha \rrangle(e)
\]

as well (as long as \(e\) is not free in \(\llangle \alpha \rrangle\); 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 (29). 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.
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 know 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:

(30) **Syntax**

\[
\begin{align*}
S & \rightarrow \text{AdvP } S \\
\text{VP} & \rightarrow \text{AdvP } \text{VP} \\
\text{VP} & \rightarrow \text{VP AdvP} \\
\text{AdvP} & \rightarrow \text{Adv}
\end{align*}
\]

(31) **Lexicon**

Adv: slowly
As we have seen above, *slowly* is interpreted as an event predicate. Its lexical entry is therefore very simple:

\[(32) \quad \text{a. } [agent] \sim \lambda e. \text{slow}(e)\]

The tree in (33) 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:
In the derivation in (33), syntactic arguments do not change the type of the verbal projections they attach to is a hallmark of
Neo-Davidsonian event semantics. The object maps a predicate of type \( \langle v, t \rangle \) (the V) to another one that is also of type \( \langle v, t \rangle \) (the VP). The subject maps a predicate of type \( \langle v, t \rangle \) (the VP) to another one that is also of type \( \langle v, t \rangle \) (the S). This is very different from what we have seen in previous chapters, where V, VP and S all had different types (namely, \( \langle e, \langle e, t \rangle \rangle \), \( \langle e, t \rangle \), and \( t \) respectively). In Neo-Davidsonian semantics, syntactic arguments are semantically indistinguishable (as far as types are concerned) from adjuncts, which map a VP of a certain type (here, \( \langle v, t \rangle \)) to another VP of the same type and which do not change the type of the VP.

10.3 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 6, 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. Sentence (34), for example, is not ambiguous. Its only reading corresponds to (35), where the event quantifier takes low scope. As for (36), that is not a possible reading of the sentence.

(34) No dog barks.

(35) a. \( \neg \exists x. \text{dog}(x) \land \exists e. \text{bark}(e) \land \text{agent}(e) = x \)
    b. “There is no barking event that is done by a dog”

(36) a. \( \exists e. \text{bark}(e) \land \neg \exists x. \text{dog}(x) \land \text{agent}(e) = x \)
b. “There is an event that is not a barking by a dog”

**Exercise 1.** How can you tell that (36) is not a possible reading of sentence (34)?

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 (35), while applying Hendriks’ object raising rule (or rather, the general schema) to the theta role head leads to the unavailable reading (36). This is shown in (37) and (38), respectively.
\[
\begin{align*}
(37) \quad S(t) \\
&\neg \exists x. \text{dog}(x) \land \exists e. \text{bark}(e) \land \\
&\text{agent}(e) = x \\
\end{align*}
\]
The interim conclusion, then, is that event semantics seems to commit us to a quantifier-raising based treatment of quantificational noun phrases.
10.3.1 A formal fragment

Let us recapitulate the additions to our fragment. The syntax is defined as in three-valued type logic ($L_\Lambda$ with three truth values as in Chapter 8), plus the following additions:

**Syntax rules.** We add the following rule:

(39)  Syntax

$$\text{DP} \rightarrow \theta \text{DP}$$

**Lexicon.** Lexical items are associated with syntactic categories as follows:

$$\theta: \quad [\text{agent}], [\text{theme}], \ldots$$

**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. $\text{bark} \leadsto \lambda e. \text{bark}(e)$
2. $\text{butter} \leadsto \lambda e. \text{butter}(e)$

Type $\langle v, e \rangle$:

1. $[\text{agent}] \leadsto \lambda x \lambda e. \text{agent}(e) = x$
2. $[\text{theme}] \leadsto \lambda x \lambda e. \text{theme}(e) = x$
10.3.2 The second strategy: Verbs as event quantifiers

In the tree in (37), 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. This means that quantificational noun phrases can be interpreted without applying quantifier raising. To implement this approach, we need to revise our semantics. The new representations for verbs are as follows:

\[
\text{(40) } \begin{align*}
\text{a. } & \text{bark} \sim \lambda f \exists e. \text{bark}(e) \land f(e) \\
\text{b. } & \text{butter} \sim \lambda f \exists e. \text{butter}(e) \land f(e) \\
\text{c. } & \ldots
\end{align*}
\]

Our grammar will continue to map verbal projections (verbs, VPs and Ss) to the same type. But this type is no longer \(<v,t>\) but \(<<v,t>,t>\). 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 look more complicated than before:

\[
\text{(41) } \begin{align*}
\text{a. } [\text{agent}] & \sim \lambda x \lambda V \lambda f. V(\lambda e. \text{agent}(e) = x \land f(e)) \\
\text{b. } [\text{theme}] & \sim \lambda x \lambda V \lambda f. V(\lambda e. \text{theme}(e) = x \land f(e)) \\
\text{c. } \ldots
\end{align*}
\]

If the root of the tree is of type \(<<v,t>,t>\), 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, \(\lambda e. \text{true}\), contains such an event:
(42)  

\[ a. \quad \lambda f \exists e [\text{bark}(e) \land \text{ag}(e) = s \land f(e)](\lambda e. \text{true}) \]

\[ b. \quad \exists e [\text{bark}(e) \land \text{ag}(e) = s \land (\lambda e. \text{true})(e)] \]

\[ c. \quad \exists e [\text{bark}(e) \land \text{ag}(e) = s \land \text{true}] \]

\[ d. \quad \exists e [\text{bark}(e) \land \text{ag}(e) = s] \]

To formalize this idea, we introduce the type-shifting rule of Quantifier Closure:

**Type-Shifting Rule 7. Quantifier Closure**

If \( \alpha \sim \llangle \alpha \rrangle \), where \( \llangle \alpha \rrangle \) is of category \( \langle (v, t), t \rangle \), then:

\[ \alpha \sim \llangle \alpha \rrangle (\lambda e. \text{true}) \]

as well.

The full derivation of the sentence is shown in (43).
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 (44). We do not need to apply quantifier raising. This is as expected, because the quantifier is contained in the entry for the verb, so the subject already takes syntactic scope over it.
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 to combine adjuncts. This makes the representations of adverbs more complicated:

(45) a. \( \textit{slowly} \sim \lambda V \lambda f. V(\lambda e. \textit{slow}(e) \land f(e)) \)

b. ... 

An example of a derivation that uses this adverb is shown in (46). To save space, the VP \textit{buttered the toast} is shown as a unit, and as before, we pretend that \textit{the toast} is a constant rather than a definite description. Nothing of consequence would change if we didn't.
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.
10.4 Conjunction in event semantics

In Chapter 9, we have seen that many uses of and can be subsumed under a general schema, discussed by Partee & Rooth (1983b) among others. This schema is repeated here:

\[
\langle \text{and} \rangle_{\tau, \langle \tau, \tau \rangle} = \begin{cases} 
\lambda q \lambda p \cdot p \land q & \text{if } \tau = t \\
\lambda X \lambda Y \lambda Z_{\sigma_1} \cdot \langle \text{and} \rangle_{\langle \sigma_2, \langle \sigma_2, \sigma_2 \rangle \rangle} (X(Z))(Y(Z)) & \text{if } \tau = \langle \sigma_1, \sigma_2 \rangle
\end{cases}
\]

What does this rule amount to in the case of VP-modifying and, as in Homer 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 (47) in each case results in the following:

\[
\begin{align*}
(48) & \quad \text{a. and}_{\text{VP}} \sim \lambda f' \lambda f \lambda e. f(e) \land f'(e) \\
& \quad \text{b. and}_{\text{VP}} \sim \lambda V' \lambda V \lambda f. V(f) \land V'(f)
\end{align*}
\]

**Exercise 2.** Show how rule (47) leads to these two representations.

As you can see in (50) and (51), these two choices lead to very different translations: (49a) and (49b) respectively.

\[
(49) \quad \begin{align*}
& \quad \text{a. } \exists e. \text{smoke}(e) \land \text{drink}(e) \land \text{agent}(e) = h \\
& \quad \text{b. } \exists e. \text{smoke}(e) \land \text{agent}(e) = h \land \\
& \quad \exists e'. \text{drink}(e') \land \text{agent}(e') = h
\end{align*}
\]

Now, (49a) cannot be the right representation of Homer smoked and drank. If this sentence is true, for all we know he might have smoked slowly and drunk quickly. In (49a) 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 (49a) sends us right back to square one. We fare much better with (49b), because it provides us with the two events we need to avoid the problem.

**Exercise 3.** Add slowly and quickly to the tree in (51) and show how the resulting formula avoids the attribution of contradictory properties to the same event.
(50)

\[ S \]
\[ t \]
\[ \exists e. \text{smoke}(e) \land \text{drink}(e) \]
\[ \land \text{agent}(e) = h \]
\[ \uparrow \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{smoke}(e) \land \text{drink}(e) \]
\[ \land \text{agent}(e) = h \]

\[ \text{DP} \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{agent}(e) = h \]
\[ \theta \]
\[ \text{DP} \]
\[ \langle e, (v, t) \rangle \]
\[ e \]
\[ \lambda x \lambda e. \text{agent}(e) = x \]
\[ \text{agent} \]
\[ \text{Homer} \]

\[ \text{VP} \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{smoke}(e) \]
\[ \land \text{drink}(e) \]
\[ \text{VP} \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{smoke}(e) \]
\[ \text{JP} \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{drink}(e) \]
\[ \text{VP} \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{drink}(e) \]

\[ \text{smoked} \]
\[ \text{smoked} \]
\[ \text{drank} \]
\[ \text{and} \]
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 pred-
icates. This is similar to DP-level conjunction, where in Chapter 9 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 *Homer smoked slowly and drank quickly.*

### 10.5 Negation in event semantics

*Note to Liz: The textbook hasn’t introduced a compositional treatment of VP-level negation at this point. I’ve created the trees but delayed writing this section until I know if we’ll have a discussion of negation somewhere earlier in the text.*
\[(52)\]

\[
\begin{align*}
S \\
\quad t \\
\exists e. \neg \text{bark}(e) \land \text{agent}(e) = s \\
\uparrow \\
\langle v, t \rangle \\
\lambda e. \neg \text{bark}(e) \land \text{agent}(e) = s
\end{align*}
\]

\[
\begin{array}{c}
\text{DP} \\
\langle v, t \rangle \\
\lambda e. \text{agent}(e) = s
\end{array}
\quad
\begin{array}{c}
\text{VP} \\
\langle v, t \rangle \\
\lambda e. \neg \text{bark}(e)
\end{array}
\]

\[
\begin{array}{c}
\theta \\
\langle e, \langle v, t \rangle \rangle \\
\lambda x \lambda e. \text{agent}(e) = x
\end{array}
\quad
\begin{array}{c}
\text{DP} \\
e \\
\langle \langle v, t \rangle, \langle v, t \rangle \rangle \\
\lambda f. f
\end{array}
\quad
\begin{array}{c}
\text{Aux} \\
\langle v, t \rangle \\
\lambda f. f
\end{array}
\quad
\begin{array}{c}
\text{VP} \\
\langle v, t \rangle \\
\lambda e. \neg \text{bark}(e)
\end{array}
\]

\[
\begin{array}{c}
\text{Neg} \\
\langle \langle v, t \rangle, \langle v, t \rangle \rangle \\
\lambda f \lambda e. \neg f(e)
\end{array}
\quad
\begin{array}{c}
\text{VP} \\
\langle v, t \rangle \\
\lambda e. \text{bark}(e)
\end{array}
\]

\[
\begin{array}{c}
\text{Spot} \\
\langle v, t \rangle
\end{array}
\quad
\begin{array}{c}
\text{bark}
\end{array}
\]

\[
\begin{array}{c}
\text{not}
\end{array}
\]
10.5.1 A formal fragment

Let us recapitulate the additions to our fragment. The syntax is defined as in three-valued type logic ($L_\lambda$ with three truth values as in Chapter 8), plus the following additions:
Syntax rules. We add the following rule:

(54) Syntax
\[ \text{DP} \rightarrow \theta \text{DP} \]

Lexicon. Lexical items are associated with syntactic categories as follows:

\[ \theta: \text{[agent], [theme], ...} \]

Translations. Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- \( e \) is \( \nu,\nu \) (as before),
- \( f \) is a variable of type \( \langle \nu, t \rangle \),
- \( V \) is a variable of type \( \langle \langle \nu, t \rangle, t \rangle \),
- bark, and butter are constants of type \( \langle \nu, t \rangle \) (as before),
- agent and theme are constants of type \( \langle \nu, e \rangle \) (as before),

The following entries replace the previous ones:
Type \( \langle \nu, t \rangle \):

1. \( \text{bark} \sim \lambda f \exists e. \text{bark}(e) \land f(e) \)
2. \( \text{butter} \sim \lambda f \exists e. \text{butter}(e) \land f(e) \)

Type \( \langle \nu, e \rangle \):

1. \( \text{[agent]} \sim \lambda x \lambda V \lambda f. V(\lambda e. \text{agent}(e) = x \land f(e)) \)
2. \( \text{[theme]} \sim \lambda x \lambda V \lambda f. V(\lambda e. \text{theme}(e) = x \land f(e)) \)

Type \( \langle \langle \langle \nu, t \rangle, t \rangle, \langle \langle \nu, t \rangle, t \rangle \rangle \rangle \):

1. \( \text{and}_{\text{VP}} \sim \lambda V' \lambda V \lambda f. V(f) \land V'(f) \)
We have introduced the following type-shifter:

**Type-Shifting Rule 8. Quantifier Closure**
If $\alpha \leadsto \langle \alpha \rangle$, where $\langle \alpha \rangle$ is of category $\langle (v, t), t \rangle$, then:

$$\alpha \leadsto \langle \alpha \rangle (\lambda e. \text{true})$$

as well.
11 | Intensional semantics

11.1 Opacity

The following might seem like a well-founded principle to adopt in everyday reasoning:

(1) **Leibniz's law**
    If two expressions have the same denotation, then if one is substituted for the other in any given sentence, the truth value of the sentence remains the same.

For example, Scott Pruitt is the currently head of the Environmental Protection Agency (EPA), and given this, it seems to follow from (2) that (3) is true.

(2) \[ \text{[Scott Pruitt]} \text{ is a climate change denier.} \]

(3) \[ \text{[The head of the EPA]} \text{ is a climate change denier.} \]

Although the latter may feel more ironic than the former, the truth value has not been changed by the substitution of the one for the other, and indeed it feels that one is licensed to conclude the latter from the former, given that Scott Pruitt is the head of the EPA.

But there are examples where Leibniz's law does not hold. For instance, the Morning Star happens to be the Evening Star. (Both name Venus.) Perhaps you did not know this already. That would not reflect poorly on your reasoning abilities, because it could have been otherwise; this is not a necessary truth. But suppose someone were to tell you that the Morning Star is the Morning Star. You
might be offended at the suggestion that you did not already know this. 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:

(4) Necessarily, the Morning Star is the Morning Star.

(5) Necessarily, the Morning Star is the Evening Star.

Contexts in which the substitution of one term for a co-referential term can lead to a change in truth value are called **referentially opaque contexts**. Our formal apparatus is currently not equipped to handle these.

Another kind of opaque context is belief. For example, if Mary does not know that Cicero is Tully, then *Mary believes that Cicero is a great orator* does not imply that *Mary believes that Tully is a great orator*. As [Quine (1956)](1956) pointed out, this opacity opens up an ambiguity which can be seen in sentences like the following:

(6) 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 interpretation is called a **de re interpretation**, where *de re* is Latin for ‘of the thing’. On another interpretation, there is no specific Republican that John believes will win; he just believes that whoever wins will be of that party. This is called a **de dicto interpretation**, where the content of the description is crucially part of the belief.

Verbs like *want, seek* and *need* also form opaque contexts, and allow for both *de re* and *de dicto* interpretations. Consider the following example:

(7) John is seeking a unicorn.

On a *de re* interpretation, there is a particular unicorn such that John is seeking it. Such an interpretation would have to be false in
the actual world, since there are (sadly) no unicorns here. But the sentence is perfectly apt to be true on a *de dicto* interpretation; it merely signals that John believes there to be unicorns (and aims to be in a state where he has found one), not that unicorns exist.

This kind of ambiguity occurs not only in philosophy texts, but also ‘in the wild’, or at least, in legal statutes. Andersen (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 1.** Consider the following case from [Andersen (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 related puzzle is that two sentences that have the same truth value can lead to different truth values when embedded under an attitude predicate like *believe* or *hope*[^1]

[^1]: Adapted from [https://mitcho.com/nus/sem2016/handout09.pdf](https://mitcho.com/nus/sem2016/handout09.pdf)
One of these sentences can be true while the other is false, even if both of the embedded sentences have the same truth value. The conclusion we are forced to draw is that the semantics of attitude verbs like believe and hope must depend not just on the truth value of the embedded clause, but on some more fine-grained aspect of its meaning. The tools we will develop in this section will allow us to capture such facts.

11.2 Modal Logic

11.2.1 Priorean tense logic

Following the presentation of Dowty et al. (1981, ch. 5), we will approach modal logic with a brief presentation of tense logic in the style developed by Arthur Prior. In this logic, a formula can vary in its truth value across time. Thus John is asleep might be true at time $t$, but false at time $t'$. A future sentence like John will be asleep can then be said to be true at time $t$ if there is a time $t'$ later than $t$ at which John 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 $T = \{ t_1, t_2, t_3 \}$, and
we have two individual constants john and mary, and one predicate constant happy. The interpretation function $I$ might then be defined as follows:

$$
I(t_1, \text{john}) = b \quad I(t_2, \text{john}) = b \quad I(t_3, \text{john}) = b \\
I(t_1, \text{mary}) = a \quad I(t_2, \text{mary}) = a \quad I(t_3, \text{mary}) = a \\
I(t_1, \text{happy}) = \{a, b, c\} \quad I(t_2, \text{happy}) = \{a, b\} \quad I(t_3, \text{happy}) = \{c\}
$$

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:

$$
\lbrack \text{happy(mary)} \rbrack_{M,g,t_1} = 1 \\
\lbrack \text{happy(mary)} \rbrack_{M,g,t_3} = 0
$$

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 $\phi$ is a formula, then $F\phi$ is a formula.
- If $\phi$ is a formula, then $P\phi$ is a formula.

($F\phi$ can be read: ‘it will be the case that $\phi$', or ‘future $\phi$’; $P\phi$ can be read: ‘it was the case that $\phi$', or ‘past $\phi$’.)

These kinds of statements can be given truth values relative to a particular time that depend on what value $\phi$ takes on at times preceding or following the evaluation time, respectively:

- $\lbrack F\phi \rbrack_{M,g,t} = 1$ iff $\lbrack \phi \rbrack_{M,g,t'} = 1$ for some $t'$ such that $t < t'$.
- $\lbrack P\phi \rbrack_{M,g,t} = 1$ iff $\lbrack \phi \rbrack_{M,g,t'} = 1$ 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 'John will have seen the report', which can take the form of $FP\phi$ 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 will discuss these further in Chapter 12 on tense. Here, it serves mainly as a warm-up for the study of the so-called 'alethic' modalities of necessity and possibility, which we turn to next.

11.2.2 Alethic logic

Just as sentences might be felt to vary in their truth value across time, so can they vary in their truth value depending on the facts. A statement like

(9) Donald Trump is President of the U.S.

expresses a truth that is certainly not a necessary truth; in principle, it is possible that Hillary Clinton could have won. In contrast, as sentence like:

(10) The President of the U.S. is President of the U.S.

is necessarily true. It could not possibly be false, in other words. In other words, the statement in (9) is CONTINGENTLY TRUE, while the statement in (10) 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.

Exercise 2. Give one 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 \( \phi \) is a formula, then \( \Box \phi \) is a formula.
- If \( \phi \) 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 \( \phi \) 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

$$I(w_1, \text{happy}) = \{a, b, c\}$$

but

$$I(w_3, \text{happy}) = \{c\}$$

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:

$$[[\text{happy(john)}]]^{M,g,w_1} = 1$$

but

$$[[\text{happy(john)}]]^{M,g,w_3} = 0$$

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. 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 synchronegorically as follows:

1. $[[\Box \phi]]^{M,g,w} = 1$ iff $[[\phi]]^{M,g,w'} = 1$ for all $w'$
2. $[[\Diamond \phi]]^{M,g,w} = 1$ 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:

---

2 See https://plato.stanford.edu/entries/possible-worlds/

3 This is one of many possibilities; this simplest system is known as S5. See Hughes & Cresswell (1968) for a fuller presentation.
• \( \Box \phi \leftrightarrow \neg \Diamond \neg \phi \)
  ‘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 \( \phi \)’

• \( \phi \rightarrow \Diamond \phi \)
  ‘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, often 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 \( \phi \), whereas \( \Box \forall x \phi \) ought to mean “in any possible world, anything that exists in that possible world is such that \( \phi \).” Similarly, \( \exists x \phi \) ought to mean that “some individual \( x \) that actually exists is in some world such that \( \phi \), 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 \( \phi \).” 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 denote a contingent property (examples from Coppock & Beaver 2015):

(11) My university email account no longer exists.

(12) If that existed, then I would have heard of it!

A famous example discussed by Russell (1905) is:

(13) 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.

Now, it is possible to combine tense logic with alethic logic, letting models contain both possible worlds and times, and having complex points of evaluation for formulas consisting of a time and a world. A formula might be true at \( \langle w_1, t_1 \rangle \) but false at \( \langle w_2, t_3 \rangle \), for example. As a broad cover-term for the points at which formulas are evaluated for truth, these pairs are called INDICES. But given that we will advocate a non-Priorean, so-called ‘referential’ theory of tense in Chapter 12, we will continue to just use possible worlds as our indices in this text.

### 11.3 Intensional Logic

#### 11.3.1 Introducing Intensional Logic

In the previous section, we defined the semantics of \( \square \) and \( \diamond \) synchroncategorematically, rather than giving \( \square \) 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 \( \square \), because the truth of a \( \square \) statement depends not on the truth value of its complement at a particular
world. We saw this above with Frege’s examples, repeated here:

(14) Necessarily, the Morning Star is the Morning Star.

(15) Necessarily, the Morning Star is the Evening Star.

Both of the embedded sentences are true, but only one of the full sentences is.

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 intension 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 $(e, t)$), 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 $\alpha$ at world $w$ (with respect to model $M$ and assignment function $g$) is denoted by $\llbracket \alpha \rrbracket_{M,g,w}$. The intension of an expression $\alpha$ is that function $f$ such that $f(w) = \llbracket \alpha \rrbracket_{M,g,w}$. That function is sometimes denoted as follows:

$$\llbracket \alpha \rrbracket_{M,g}$$

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 $\square$ 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 $\alpha$ a new expression denoting the intension of $\alpha$. 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 $\alpha$. For example, the formula

$$\text{happy}(m)$$

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:

\[ \hat{\text{happy}}(m) \]

has the intension of \( \text{happy}(m) \) as its extension. (Put more simply: The extension of \( \hat{\text{happy}}(m) \) is the intension of \( \text{happy}(m) \).) 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 extension of 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 \( \tau \), a new type \( \langle s, \tau \rangle \). The complete type system is now as follows:

- \( t \) is a type
- \( e \) is a type
- If \( \sigma \) and \( \tau \) are types, then so is \( \langle \sigma, \tau \rangle \)
- If \( \tau \) is any type, then \( \langle s, \tau \rangle \) is a type.

Our syntax rules will be extended so that if \( \alpha \) is an expression of type \( \tau \), 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 \( D_\tau \), where \( D_\tau \) is the domain of entities denoted by expressions of type \( \tau \). The official semantic rule for \( \hat{\cdot} \) is as follows:

- If \( \alpha \) is an expression of type \( \tau \), then \( \llbracket \hat{\alpha} \rrbracket^{M,g,w}_{s,\tau} \) is that function \( f \) with domain \( W \) such that for all \( w \in W \): \( f(w) = \llbracket \alpha \rrbracket^{M,g,w} \).
The hat operator also has a partner, \( \checkmark \) which moves from intensions to extensions. If \( \alpha \) is an expression of type \( \langle s, \tau \rangle \), then \( \checkmark \alpha \) is an expression of type \( \tau \). Its semantics is defined as follows:

- If \( \alpha \) is an expression of type \( \langle s, \tau \rangle \), then \( \hat{\alpha}^M,g,w \) is the result of applying the function \( \hat{\alpha}^M,g,w \) to \( w \).

With these tools in hand, let us now consider how we might get a handle on the \textit{de dicto} / \textit{de re} ambiguity and related puzzles. Let us introduce a constant \( \text{bel} \), 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 \text{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 \textit{de dicto} reading of a sentence like \textit{John believes that a Republican will win} can then be represented as follows:

\[ \text{bel}(\text{john}, \checkmark \exists x[\text{repub}(x) \land \text{win}(x)]) \]

The \textit{de re} reading can be represented:

\[ \exists x[\text{repub}(x) \land \text{bel}(\text{john}, \checkmark[\text{win}(x)])] \]

In the latter formula, the existential quantifier and the predicate \text{repub} occur outside the scope of the belief operator. So on the \textit{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 \textit{de dicto} reading, on which the content of John’s belief involves that property.

Let us consider one more example of a \textit{de dicto} / \textit{de re} ambiguity, this time involving a proper name. The following sentence can be understood in two ways:

(16) John believes that Miss America is bald.
On the *de re* interpretation, John might believe of some individual who happens to be Miss America without John knowing, that she is bald. On the *de dicto* interpretation, John might not have any acquaintance at all with the individual who is Miss America, but believe that, whoever they are, they are bald. The two interpretations can be represented as follows:

(17)  \[ \lambda x[\text{bel}(\text{john}, \uparrow \text{bald}(x))](m) \]

(18)  \[ \text{bel}(\text{john}, \uparrow \text{bald}(m)) \]

The former captures the *de re* interpretation; the latter captures the *de dicto* one. When \(m\) is in the scope of the \(\text{bel}\) 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.

11.3.2  **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.

11.3.2.1  **Semantics**

The types are defined recursively as follows:

- \(t\) is a type
- \(e\) is a type
- If \(\sigma\) and \(\tau\) are types, then so is \(\langle \sigma, \tau \rangle\)
• If \( \tau \) is any type, then \( \langle s, \tau \rangle \) is a type.

The set of expressions of type \( \tau \), for any type \( \tau \), is defined recursively as follows:

1. **Basic Expressions**  
   For each type \( \tau \),
   
   (a) the **non-logical constants** of type \( \tau \) are they symbols of the form \( c_{n,\tau} \) for each natural number \( n \).
   
   (b) the **variables** of type \( \tau \) are the symbols of the form \( v_{n,\tau} \) for each natural number \( n \).

2. **Predication**  
   For any types \( \sigma \) and \( \tau \), if \( \alpha \) is an expression of type \( \langle \sigma, \tau \rangle \) and \( \beta \) is an expression of type \( \tau \) then \( \alpha(\beta) \) is an expression of type \( \tau \).

3. **Equality**  
   If \( \alpha \) and \( \beta \) are terms, then \( \alpha = \beta \) is an expression of type \( t \).

4. **Negation**  
   If \( \phi \) is a formula, then so is \( \neg \phi \).

5. **Binary Connectives**  
   If \( \phi \) and \( \psi \) are formulas, then so are \( \neg \phi, [\phi \land \psi], [\phi \lor \psi], [\phi \rightarrow \psi] \), and \( [\phi \leftrightarrow \psi] \).

6. **Quantification**  
   If \( \phi \) 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 \( \alpha \) is an expression of type \( \tau \) and \( u \) is a variable of type \( \sigma \) then \( [\lambda u. \alpha] \) is an expression of type \( \langle \sigma, \tau \rangle \).

8. **Alethic modalities** (**new!**)  
   If \( \phi \) is a formula, then \( \Box \phi \) and \( \Diamond \phi \) are formulas.
9. **Intensionalization** (new!)
   if $\alpha$ is an expression of type $\tau$, then $\hat{\alpha}$ is an expression of type $(s, \tau)$.

10. **Extensionalization** (new!)
    If $\alpha$ is an expression of type $(s, \tau)$, then $\check{\alpha}$ is an expression of type $\tau$.

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 $b$, $D_{(a,b)}$ 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 $\alpha$ is a non-logical constant, then $[\alpha]^{M,g} = I(\alpha)$.
   
   (b) If $\alpha$ is a variable, then $[\alpha]^{M,g} = g(\alpha)$.

2. **Predication**
   If $\alpha$ is an expression of type $(a, b)$, and $\beta$ is an expression of type $a$, then $[\alpha(\beta)] = [\alpha](\[\beta\])$.

3. **Equality**
   If $\alpha$ and $\beta$ are terms, then $[\alpha = \beta]^{M,g} = 1$ iff $[\alpha]^{M,g} = [\beta]^{M,g}$. 
4. Negation
If $\phi$ is a formula, then $\llbracket \neg \phi \rrbracket^{M,g} = 1$ iff $\llbracket \phi \rrbracket^{M,g} = 0$.

5. Binary Connectives
If $\phi$ and $\psi$ are formulas, then:

(a) $\llbracket \phi \land \psi \rrbracket^{M,g} = 1$ iff $\llbracket \phi \rrbracket^{M,g} = 1$ and $\llbracket \psi \rrbracket^{M,g} = 1$.
(b) $\llbracket \phi \lor \psi \rrbracket^{M,g} = 1$ iff $\llbracket \phi \rrbracket^{M,g} = 1$ or $\llbracket \psi \rrbracket^{M,g} = 1$.
(c) $\llbracket \phi \rightarrow \psi \rrbracket^{M,g} = 1$ iff $\llbracket \phi \rrbracket^{M,g} = 0$ and $\llbracket \psi \rrbracket^{M,g} = 1$.
(d) $\llbracket \phi \leftrightarrow \psi \rrbracket^{M,g} = 1$ iff $\llbracket \phi \rrbracket^{M,g} = \llbracket \psi \rrbracket^{M,g}$.

6. Quantification

(a) If $\phi$ is a formula and $v$ is a variable of type $a$ then $\llbracket \forall v. \phi \rrbracket^{M,g} = 1$ iff for all $k \in D_a$:

\[
\llbracket \phi \rrbracket^{M,g}[v \mapsto k] = 1
\]

(b) If $\phi$ is a formula and $v$ is a variable of type $a$ then $\llbracket \exists v. \phi \rrbracket^{M,g} = 1$ iff there is an individual $k \in D_a$ such that that:

\[
\llbracket \phi \rrbracket^{M,g}[v \mapsto k] = 1.
\]

7. Lambda Abstraction
If $\alpha$ is an expression of type $a$ and $u$ a variable of type $b$ then $\llbracket \lambda u. \alpha \rrbracket^{M,g}$ is that function $h$ from $D_b$ into $D_a$ such that for all objects $k$ in $D_b$, $h(k) = \llbracket \alpha \rrbracket^{M,g}[u \mapsto k]$.

8. Alethic modalities (new!)

(a) $\llbracket \Box \phi \rrbracket^{M,g,w} = 1$ iff $\llbracket \phi \rrbracket^{M,g,w'} = 1$ for all $w'$
(b) $\llbracket \Diamond \phi \rrbracket^{M,g,w} = 1$ iff $\llbracket \phi \rrbracket^{M,g,w'} = 1$ for some $w'$

9. Intensionalization (new!)
If $\alpha$ is an expression of type $\tau$, then $\llbracket \hat{\alpha} \rrbracket^{M,g,w}$ is that function $f$ with domain $W$ such that for all $w \in W$: $f(w)$ is $\llbracket \alpha \rrbracket^{M,g,w}$. 
10. **Extensionalization (new!)**

If $\alpha$ is an expression of type $\langle s, \tau \rangle$, then $\sem{\alpha}^{M, g, w}$ is the result of applying the function $\sem{\alpha}^{M, g, w}$ to $w$.

**Exercise 3.** Formulate a lexical entry for a necessity operator, and show how it accounts for the examples involving the Morning Star and the Evening Star.

**Exercise 4.** Consider (8) in light of the presentation just given. Give logical translations for the two sentences involved, and explain how they can differ in their truth value even when the embedded statements have the same truth value, by giving an example model where this is the case.

**Exercise 5.** Formalize the *de dicto* and *de re* readings of the what the statute prohibits in the Andersen example on page 315 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 6.** Is it possible to give a non-syncategorematic treatment of the hat operator $\hat{\cdot}$? Explain why or why not.

### 11.4 Fregean sense and hyperintensionality

Frege's assessment of his puzzle about identity built on a distinction between *sense* and *reference*. For Frege, the expressions *the Morning Star* and *the Evening Star* 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 Toida’s conjecture. But from (19), it does not follow that (20) is true.

(19) John believes that $2 + 2 = 4$.
(20) John believes that Toida’s conjecture is true.

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 (21) does not imply (22) (example from Muskens 2005):

(21) John believes that the cat is in if the dog is out.
(22) John 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 hyperintensionality 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.

### 11.5 Explicit quantification over worlds

The astute reader may have noticed that in the system described in §11.3 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

$$\overset{\sim}{\text{bald}}(m)$$

one writes something along the lines of:

$$\lambda w.\text{bald}_w(m)$$

where the subscript on $w$ is meant to indicate that $\text{const}$ denotes a function that takes a possible world as an argument, in addition to an individual. A famous example of this sort of system is Gallin's (1975) $\text{Ty}_2$, 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.
11.5.1 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.

11.6 Logic of Indexicals

We are now in a position to appreciate Kaplan’s (1977) theory of indexicals, published in a famous article called Demonstratives. 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]”)

What is particularly interesting about indexicals is that they exhibit what Kaplan calls a ‘deviant logic’. To illustrate, consider the following sentence:

(23) I am here now.
This sentence is similar to a tautology in that it is true whenever it is uttered (given normal interpretations of the terms involved; here we are setting aside answering machine cases where *I am not here right now* is in fact perfectly apt to be true). But (23) is not a necessary truth, because the circumstances could be otherwise. Any given speaker of such a sentence might well have ended up somewhere else that day.

This puzzle can be solved with the help of two levels of meaning, one which he calls **Character** and another which he calls **Content**. He bases this conclusion on the following two principles:

- The referent of a pure indexical depends on the context, and the referent of a demonstrative depends on the associated demonstration.

- Indexicals, pure and demonstrative alike, are directly referential.

Kaplan defines **directly referential** as follows: an expression is directly referential if its referent, once determined, is taken as fixed for all possible circumstances. By **circumstances**, he means possible worlds (or indices, more generally). So a directly referential expression would have a constant intension, i.e., one whose value does not change depending on the input possible world. **Kripke (1980)** used the term **rigid designator** for more or less the same concept. On Kripke’s view, proper names like *John* are rigid designators, and definite descriptions like *the president* are not.

For example, consider the following sentence:

(24) The president is a Democrat.

This sentence is false in the actual world. But there is an alternative world where Hillary is president (and still a Democrat); relative to that world, *the president* picks her out and the sentence
Intensional semantics

is true. What affects the truth of the sentence at a given world is whether or not whoever happens to be president in that world is a Democrat at that world. But suppose we try:

(25) Donald Trump is a Democrat.

This sentence is also false in the actual world. But if we move to an alternative world where Hillary is president (and still a Democrat), the sentence does not suddenly become true at that world. Of course, the sentence could be true, if Donald Trump himself is a Democrat in that world. But who is president does not matter. What matters to the truth of this sentence is that Donald Trump himself has the property of being a Democrat.

Now suppose that Donald Trump decides to refer to himself with a first-person pronoun rather than in the third person, and utters:

(26) I am a Democrat.

Again, the sentence is false in this world. And its ‘modal profile’ as it were is exactly the same as the one for the proper name. If we move to another world where Hillary Clinton is a president (and still a Democrat), the sentence does not necessarily become true. What matters is whether Trump himself is a Democrat. So:

- *Donald Trump* designates the same individual in every possible world; this expression is directly referential.

- *The president* can designate different individuals in different possible worlds.

- When Donald Trump says *I*, he means *Donald Trump*. *I* is directly referential too.

(A slight complication comes from the fact that there are so-called descriptive uses of indexicals, as [Nunberg (1992)](Nunberg1992) discusses. His example is the following, spoken by a person on death row:

(27) I am traditionally allowed a last meal.

But let us ignore this fact for the time being.

Recall that an expression is called ‘directly referential’ if its referent, once determined, is taken as fixed for all possible circumstances, and Kaplan claims that *I* has this property. However, as Kaplan continues, “This does not mean it could not have been used to designate a different object; in a different context, it might have. But regardless of the circumstance of evaluation, it picks out the same object.” For example, if Hillary Clinton says *I am a Democrat*, then *I* picks out Hillary Clinton rather than Donald Trump.

It therefore becomes important to distinguish between the following two concepts:

- **CONTEXT OF UTTERANCE**: Who is speaking to whom, where, when, what they are gesturing to, etc.

- **CIRCUMSTANCE OF EVALUATION**: A possible world at which the truth of the utterance might be evaluated.

The word *I* picks out the same individual in every *circumstance of evaluation*, but different individuals in different *contexts of utterance*. Whereas the meaning of the indexical *I* is fixed across all circumstances of evaluation, but variable across contexts of use, the meaning of a definite description is variable across circumstances of evaluation (and arguably variable across contexts of use as well).

Now consider the following two utterances:

(28) a. (May 11, 2010, uttered by Elizabeth Coppock:) I am turning 30 today.

b. (May 12, 2010, uttered by Elizabeth Coppock:) I am turning 30 today.

Do they have the same meaning or different meaning? How about the following pair:
Intuitions differ.

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.

- **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 (28) have the same character, while the pair in (29) 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.)

Given a particular context, the contribution of an indexical to the content of a sentence is fixed, even though an indexical does not always refer to the same thing. In some sense, the indexical *I* means something like ‘the speaker of the sentence’, and in that sense, it has some sort of descriptive meaning. But, Kaplan says
(p. 498), “this meaning is relevant only to determining a referent in a context of use and not to determining a relevant individual in a circumstance of evaluation.” In other words, the descriptive meaning is part of the character, but not the content.

Imagine if it were otherwise! Kaplan writes,

Suppose *I do not exist* is true in a circumstance of evaluation if and only if the speaker (assuming there is one) of the circumstance does not exist in the circumstance. Nonsense! If that were the correct analysis, what I said could not be true. From which it follows that:

*It is impossible that I do not exist.*

In other words, taking the descriptive content to be part of the content, rather than the character, leads to an absurd conclusion. Kaplan’s theory does not.

Similarly, Kaplan’s theory can explain why *I am here now* is a **LOGICAL TRUTH**, in the sense that whenever it is uttered, it is true, yet not a **NECESSARY TRUTH**, because the circumstances could be otherwise. This is unusual: Normally logical truths are necessary truths! If $\phi$ is true in all models, then $\square \phi$ is true in all models. So as Kaplan (1999) describes it, indexicals produce “a distinctive and deviant pattern of logical consequence”.

So, how does it work? Very simply. All that is needed in order to capture this deviant logic is to add context as a parameter of evaluation. So now, in addition to a model $M$, an assignment function $g$, and a world $w$, we will also determine the extension of expressions relative to a context of utterance $c$. The extension of an expression $\alpha$ is denoted as follows:

$$\llbracket \alpha \rrbracket^{M,g,w,c}$$

The $M$, $g$ and $w$ parameters work just as we have already seen. For example, if $\alpha$ is a variable, then $\llbracket \alpha \rrbracket^{M,g,w,c} = g(\alpha)$. The modal op-
Intensional semantics

Operators are also defined just as before: $\left\lbrack \square \phi \right\rbrack_M, g, w, c = 1$ iff $\left\lbrack \phi \right\rbrack_M, g, w', c = 1$ for all $w'$.

The context parameter $c$ represents the context of utterance. It determines a speaker $sp(c)$, an addressee $ad(c)$, a time of utterance $t(c)$, a location of utterance $l(c)$, and a designated circumstance of evaluation $w(c)$. The character can be defined formally by abstracting over the context parameter, so in general, the character of an expression $\alpha$ will be a function $f$ such that for any $c$, $f(c) = \left\lbrack \alpha \right\rbrack_M, g, w, c$.

The indexical constants $i$ ‘I’, $u$ ‘you’ $n$ ‘now’, and $h$ ‘here’ can be defined as follows:

(30) $\left\lbrack i \right\rbrack_M, g, w, c = sp(c)$
(31) $\left\lbrack u \right\rbrack_M, g, w, c = ad(c)$
(32) $\left\lbrack n \right\rbrack_M, g, w, c = t(c)$
(33) $\left\lbrack h \right\rbrack_M, g, w, c = l(c)$

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 $\phi$ in $c$ is true iff the content expressed by $\phi$ 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 a simplest 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 \( \text{exists} \) at the world of the context. Formally:

\[(34) \text{ If } c \in C, \text{ then } sp(c) \in I_{w(c)}(\text{exists}).\]

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 \( \text{exists} \) predicate in the world of \( c \) contains the speaker of \( c \). This condition guarantees that the character of ‘I exist’, or, formally \( \text{exists}(i) \) 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. But it is not a necessary truth. The content expressed is one that is contingent.

Otherwise, Kaplan’s logic of indexicals is just like IL. In particular, Kaplan’s logic of indexicals contains necessity and possibility operators defined in the standard way. So \( \square \text{exists}(i)^{M,g,w,c} = 1 \) iff \( \text{exists}(i)^{M,g,w',c} = 1 \) for all \( w' \). If, relative to \( c \), \( i \) denotes an individual that does not exist in every world, then \( \square \text{exists}(i) \) will be false. Thus \( \square \text{exists}(i) \) is neither a logical truth nor a necessary truth.

**Exercise 7.** Explain why \( \text{I do not exist} \) is logically false yet not necessarily false in this framework.

**Exercise 8.** 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?

---

\(^4\text{Cf. conditions 10 and 11, p. 544 of Kaplan (1977).}\)
Kaplan's framework captures the fact that words like *possible* and *necessary* do not affect the interpretation of indexicals. As Kaplan (1977, 499) puts it, “See how rigidly the indexicals cling to the referent determined in the context of use:”:

(35) It is possible that in Pakistan, in five years, only those who are actually here now are envied.

Moving to another circumstance of evaluation with *it is possible that*, another location with *in Pakistan*, and another time with *in five years* does not change the interpretation of *actually*, *here*, and *now*. So these operators may shift the circumstance of evaluation, but they don't shift the context.

Now, one might ask if there are operators that are similar to \(\Box\) except that they quantify over different possible contexts of utterance rather than different possible circumstances of evaluation. Kaplan asks this question in the following way (p. 510):

Are there such operators as 'In some contexts it is true that', which when prefixed to a sentence yields a truth if and only if in some context the contained *sentence* (not the context expressed by it) expresses a content that is true in the circumstances of that context? Let us try it:

(36) In some contexts it is true that I am not tired now.

For [(36)] to be true in the present context it suffices that some [speaker] of some context not be tired at the time of that context... I am not saying we could not construct a language with such operators, just that English is not one. And such operators *could not be added to it*.

He continues,
There is a way to control an indexical, to keep it from taking primary scope, and even to refer it to another context (this amounts to changing character). Use quotation marks. If we mention the indexical rather than use it, we can, of course, operate directly on it. Carnap once pointed out to me how important the difference between direct and indirect quotation is in:

Otto said “I am a fool.”
Otto said that I am a fool.

Operators like ‘In some context it is true that’, which attempt to meddle with character, I call monsters. I claim that none can be expressed in English (without sneaking in a quotation device).

As it turned out, some years later, monsters were found in the Semitic language Amharic (Schlenker, 2003), where the word-by-word translation of John said that I am a hero means ‘John said that he was a hero’, and it can be shown that quotation is not involved. So apparently monsters do exist after all, and they have proved quite rewarding to study.

Let us take stock. One way of looking at what Kaplan achieved in Demonstratives is as “the scientific realization of a Strawsonian semantics of use.” Strawson was one of the philosophers that protested against the logical approach to natural language semantics under the slogan, ‘meaning is use’. Here is a representative quotation from Strawson (1950):

Meaning (in at least one important sense) is a function of the sentence or expression; mentioning and referring and truth or falsity, are functions of the use of the sentence or expression. To give the meaning of an expression (in the sense in which I am using the word) is to give general directions for its use to refer to or mention particular objects or persons; to give the
meaning of a sentence is to give general directions for its use in making true or false assertions.

Kaplan (1999), reflecting on *Demonstratives*, says the following:

In [Demonstratives], I tried to show that by adding context as a parameter, Strawson’s “conventions for referring” as he calls them, even if neglected by logicians, could be accommodated within the range of our methods. And at the time I regarded my work as extending current semantical methods just to the degree necessary to incorporate the indexicals. I regarded what I was doing as a sort of epicycle on Carnap’s method of extension and intension and I didn’t think of it as involving any different conception of semantics or what semantics was supposed to do.

Some years ago, it occurred to me that the analysis of indexicals in Demonstratives could be seen as the scientific realization of a Strawsonian semantics of use. *Ask not after other-worldly meanings, ask only after rules of use.* [Emphasis added]
12  Tense

In which we give a theory of tense.
Bibliography


Coppock, Elizabeth. under second-round review. Outlook-based semantics. Submitted to *Linguistics and Philosophy*.


Gallin, Daniel. 1975. *Intensional and higher order modal logic*. Amsterdam: North Holland Press.


Kaplan, David. 1999. The meaning of *ouch* and *oops*. Lecture presented at the University of California at Berkeley.


Oliver, Alex & Timothy Smiley. 2013. *Plural logic*. Oxford University Press.


