Invitation to Formal Semantics

(Formerly known as Semantics Boot Camp)

Elizabeth Coppock
&
Lucas Champollion

Draft of August 20, 2019
Preface

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

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

One distinguishing feature of this book is the Lambda Calculu-
*lator, an interactive, graphical application to help students practice derivations in the typed lambda calculus. It is designed for both students and teachers, with modules for online classroom instruction, graded homework assignments, and self-guided practice. The primary function is to assist in the computation of natural language denotations up a syntactic tree. To this end, the program detects common errors and attempts to provide intelligent feedback to the student user and a record of performance for the instructor. Many exercises in this textbook are designed to be solved with the Lambda Calculator. The software runs on Mac, Linux, and Windows machines. The student version of the calculator is available as a free download from [www.lambdacalculator.com](http://www.lambdacalculator.com), which also provides documentation and exercise files; the teacher edition, which offers advanced functionality, is available to instructors on request by writing to champollion@nyu.edu The Lambda Calculator was originally developed by Lucas Champollion, Maribel Romero, and Josh Tauberer (Champollion et al., 2007). Further contributions to its code and documentation have been made by Anna Alsop, Dylan Bumford, Raef Khan, and Alex Warstadt, whose help we gratefully acknowledge.

Instructors who have previously taught from Heim & Kratzer (1998) will find much familiar material in this book, such as the composition rules: Functional Application, Predicate Modification, Predicate Abstraction, Lexical Terminals, and the Pronouns and Traces Rule. The most prominent difference in the framework is that we translate English expressions into well-formed expressions of lambda calculus rather than specifying denotations directly using an informal metalanguage containing lambdas. Our style of analysis involves defining a formal *representation language*, which is a logic with a syntax and a semantics (the language of lambda calculus, with some enhancements borrowed from the linguistic tradition), and defining a systematic *translation* from English to that language (‘translate-first, interpret-second’, in slogan form). Our logic-based representation language is both more
precise and more compact than the informal language based on paraphrases adopted in [Heim & Kratzer (1998)]. Our derivations easily fit into tree representations. Here is a sample derivation involving both Predicate Modification and Functional Application:

```
NP
  e
  ιx. [Textbook(x) ∧ On(x, sem)]
```

```
D
  ⟨⟨e, t⟩, e⟩
  λP. ιx. P(x)
  the

N'
  ⟨e, t⟩
  λx. [Textbook(x) ∧ On(x, sem)]

N
  ⟨e, t⟩
  λx. Textbook(x)
  textbook

PP
  ⟨e, t⟩
  λx. On(x, sem)

P
  ⟨e, ⟨e, t⟩⟩
  λy λx. On(x, y)
  on

NP
  e
  sem
```

Another important departure from the [Heim & Kratzer (1998)] framework is in the treatment of presupposition. Partial functions are replaced with total functions whose range includes an ‘undefined’ value, and the partiality operator from [Beaver & Krahmer (2001)] is introduced. This means that Functional Application is always defined, and it is easy to read off the presuppositions of a sentence from its logical translation, and definedness conditions do not get lost along the way.

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

We are grateful to Omar Agha, Masha Esipova, and Alicia Parrish for assisting us with preparing the text of this book and for creating the answer keys.
# Contents

1 Introduction .................................................................................................................. 13  
1.1 Entailment .................................................................................................................. 13  
  1.1.1 Entailment vs. implicature ................................................................................... 18  
  1.1.2 Entailment vs. presupposition ............................................................................ 26  
  1.1.3 Entailment and validity ....................................................................................... 31  
1.2 Framework .................................................................................................................... 33  
  1.2.1 Model-theoretic semantics .................................................................................. 33  
  1.2.2 Indirect interpretation ......................................................................................... 36  
  1.2.3 Compositional semantics .................................................................................... 38  

2 Sets, Relations, and Functions ......................................................................................... 41  
2.1 Introduction .................................................................................................................. 41  
2.2 Sets ................................................................................................................................ 41  
2.3 Ordered pairs and relations ........................................................................................ 53  
  2.3.1 Ordered pairs ...................................................................................................... 53  
  2.3.2 Relations ............................................................................................................ 54  
2.4 Functions ...................................................................................................................... 58  
2.5 Negative polarity items ............................................................................................... 64  

3 First-order logic .................................................................................................................. 75  
3.1 Introduction .................................................................................................................. 75  
3.2 Propositional logic ....................................................................................................... 76  
  3.2.1 Formulas and propositional letters ...................................................................... 78  
  3.2.2 Boolean connectives ............................................................................................ 81  
  3.2.3 Conditionals and biconditionals ....................................................................... 92  

7
## CONTENTS

5.2.6  *Agneta is a singer*  ........................................ 183

5.3  Quantifiers: A sneak peek  ..................................... 184

5.3.1  Toy fragment  .................................................. 189

6  Other composition rules  .......................................... 193

6.1  Predicate Modification  ........................................... 193

6.1.1  *Anni-Frid is a Norwegian millionaire*  .................. 193

6.2  Relative clauses  .................................................. 202

6.3  Pronouns  .......................................................... 212

7  Quantifiers  .......................................................... 223

7.1  Quantifiers: Type $\langle e, t \rangle$  ................................. 223

7.1.1  Quantifiers: not type $e$  ...................................... 223

7.1.2  Solution: Quantifiers  .......................................... 228

7.2  Generalized Quantifiers  ......................................... 235

7.3  Quantifiers in object position  ................................... 235

7.3.1  Quantifier Raising  ............................................. 235

7.3.2  A type-shifting approach  ..................................... 242

7.3.3  Putative arguments for the movement analysis  ............ 248

7.4  Quantifiers, negation and adverbs  ............................... 256

8  Presupposition  ...................................................... 261

8.1  Introduction  ....................................................... 261

8.2  The definite article  .............................................. 263

8.3  Definedness conditions  .......................................... 274

8.4  Projection problem  ................................................ 282

9  Dynamic semantics  .................................................. 285

9.1  Introduction  ........................................................ 285

9.2  Presupposition in dynamic semantics  ............................ 285

9.3  Pronouns with indefinite antecedents  ........................... 295

9.4  File change semantics  ............................................. 302

9.5  Discourse Representation Theory  ................................. 306

9.6  Compositional DRT  ............................................... 314
## 10 Coordination and Plurals

10.1 Coordination ................................................. 329
10.2 Mereology .................................................. 336
10.3 The plural .................................................. 340
   10.3.1 Algebraic closure ................................... 340
   10.3.2 Plural definite descriptions ....................... 342
10.4 Cumulative readings ...................................... 345
10.5 Formal mereology ......................................... 347
10.6 A formal fragment ........................................ 350
   10.6.1 Logic syntax ......................................... 350
   10.6.2 Logic semantics ..................................... 351
   10.6.3 English syntax ....................................... 351
   10.6.4 Translations .......................................... 352

## 11 Event semantics

11.1 Why event semantics ..................................... 355
   11.1.1 The Neo-Davidsonian turn ......................... 359
11.2 Composition in Neo-Davidsonian event semantics .... 361
   11.2.1 Verbs as predicates of events .................... 362
   11.2.2 A formal fragment .................................. 367
11.3 Quantification in event semantics ..................... 368
   11.3.1 Verbs as event quantifiers ....................... 372
   11.3.2 Another formal fragment ......................... 378
11.4 Conjunction in event semantics ....................... 379
11.5 Negation in event semantics ........................... 384

## 12 Tense

12.1 Introduction .............................................. 387
12.2 Aspect ..................................................... 388
   12.2.1 Aktionsart .......................................... 388
   12.2.2 Viewpoint aspect .................................. 390
12.3 Indexicality ............................................... 395
12.4 Tense ...................................................... 400
   12.4.1 Priorean tense logic ............................... 400
   12.4.2 Shortcomings of the Priorean theory of tense ... 402
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.5</td>
<td>A formal theory of tense</td>
<td>404</td>
</tr>
<tr>
<td>12.5.1</td>
<td>Anaphoric theory of the past</td>
<td>404</td>
</tr>
<tr>
<td>12.6</td>
<td>Future (in English)</td>
<td>411</td>
</tr>
<tr>
<td>12.6.1</td>
<td>Sequence of tense</td>
<td>412</td>
</tr>
<tr>
<td>13</td>
<td>Modality</td>
<td>415</td>
</tr>
<tr>
<td>13.1</td>
<td>Introduction</td>
<td>415</td>
</tr>
<tr>
<td>13.2</td>
<td>Opacity</td>
<td>415</td>
</tr>
<tr>
<td>13.3</td>
<td>Modal Logic</td>
<td>421</td>
</tr>
<tr>
<td>13.3.1</td>
<td>Alethic logic</td>
<td>421</td>
</tr>
<tr>
<td>13.4</td>
<td>Intensional Logic</td>
<td>427</td>
</tr>
<tr>
<td>13.4.1</td>
<td>Introducing Intensional Logic</td>
<td>427</td>
</tr>
<tr>
<td>13.4.2</td>
<td>Formal fragment</td>
<td>432</td>
</tr>
<tr>
<td>13.5</td>
<td>Fregean sense and hyperintensionality</td>
<td>436</td>
</tr>
<tr>
<td>13.6</td>
<td>Explicit quantification over worlds</td>
<td>437</td>
</tr>
<tr>
<td>13.6.1</td>
<td>Modal auxiliaries</td>
<td>438</td>
</tr>
<tr>
<td>13.7</td>
<td>Indexicals and Necessity</td>
<td>439</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
<td>443</td>
</tr>
<tr>
<td>A.1</td>
<td>Logic: Partial typed lambda calculus (L₃)</td>
<td>444</td>
</tr>
<tr>
<td>A.1.1</td>
<td>Syntax of L₃</td>
<td>444</td>
</tr>
<tr>
<td>A.1.2</td>
<td>Semantics of L₃</td>
<td>445</td>
</tr>
<tr>
<td>A.2</td>
<td>Syntax of English fragment</td>
<td>447</td>
</tr>
<tr>
<td>A.3</td>
<td>Translations</td>
<td>449</td>
</tr>
<tr>
<td>A.3.1</td>
<td>Lexical entries</td>
<td>449</td>
</tr>
<tr>
<td>A.3.2</td>
<td>Composition rules</td>
<td>452</td>
</tr>
</tbody>
</table>
1 | Introduction

1.1 Entailment

This is a book about 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 about the actor Hugh Laurie that says:

(1) He speaks French, German and Spanish.

From this statement, a reader would likely infer that Hugh Laurie does not speak Cantonese—if he did, then Cantonese would have been listed among the languages he speaks. (Interestingly, the same does not apply to English!) The sentence would not be false, strictly speaking, if it turned out that Hugh Laurie also spoke Cantonese, but still, it somehow implies that he does not. Another inference that a reader would be licensed to draw is that Hugh Laurie speaks more than two languages. This conclusion necessarily follows from the sentence. But if we asked Google whether Hugh Laurie speaks Cantonese, or whether he speaks more than two languages, Google would not be able to tell us the answer.

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, logically or otherwise. The more implication relations a theory correctly captures, the better it is. (Semantic theories may be motivated by other kinds of data as well, including intuitions about which sentences sound acceptable vs. odd, which sentences are inherently contradictory or inherently true, and facts about frequency in natural language corpora.)

**Exercise 1.** What is the primary kind of data that is used in the study of semantics? What other kinds of data can be used?

Semanticists tend to use the word ‘imply’, and its nominal form, ‘implication’, in a very broad sense. In this broad sense, (1) implies both that Hugh Laurie doesn’t speak Cantonese and that he speaks more than two languages, although these are two different kinds of implications. We will not attempt to define what it means for one sentence to imply another, except to say that we mean it in a sense that covers a wide range of subtypes of implication, including ENTAILMENTS, IMPLICATURES, and PRESUPPOSITIONS. This chapter provides a brief introduction to all three.

When one sentence logically implies another, we say that the first ENTAILS the other (nominal form: entailment). For example, (1) entails *Hugh Laurie speaks more than two languages*, because the latter follows logically from the first. What it means to ‘follow logically’ can be made more precise in the following terms:

---

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) A sentence A entails a sentence B if and only if:
Whenever A is true, B is true too.

Here, *whenever* means something like ‘in every possible world or situation where’. For example, in every possible world or situation where *Hugh Laurie speaks French, German and Spanish* is true, *Hugh Laurie speaks more than two languages* is true too.

Here are some more examples of entailments. In the following pairs, the (a) sentence entails the (b) sentence:

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

(4) a. Mary invited both Fred and Jack.
   b. Mary invited Fred.

   b. Jane graduated from MIT.

(6) 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 certain basic assumptions about the world. For example, given that all sportscars are cars, *All cars are blue* entails *All sportscars are blue*, because in every possible 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.

The definition of entailment just given appeals to the notion of ‘possible world or situation’. By this, we mean something like ‘conceivable alternative states of affairs’ or ‘ways the world could be’. For example, in the actual world, Madonna became a pop star. But what if she had become an astronaut instead? That possibility holds in alternative possible worlds. There are many possible worlds in which Madonna became an astronaut. In some of them, the iPhone was not invented until 2050, and in others, the iPhone
was never invented. In some of them, Barack and Michelle Obama met each other in law school, and in others, they never even met. So-called ‘modal realists’ hold that possible worlds actually exist in the sense that they occupy space and time. We are not committed to this view of possible worlds by any means; we just need a way to talk about how the world would be if such-and-such were the case, and we use possible worlds for this purpose.

In the logical tradition founded by Gottlob Frege, Alfred Tarski, Rudolph Carnap, and Richard Montague, the basic meaning of a sentence is its truth conditions: the conditions under which the sentence is true. Clearly, knowing the meaning of a sentence does not require knowing whether the sentence is in fact true, but it does require being able to discriminate between situations/worlds in which the sentence is true and situations/worlds in which it is false. Truth-conditional semantics provides a systematic association between sentences and their truth conditions.

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 + 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 some important aspects of meaning.
Among the aspects of meaning that truth conditions can capture is entailment: If the truth conditions for B are satisfied whenever the truth conditions for A are satisfied, then A entails B. And propositions are one way of characterizing truth conditions. The set of all possible worlds in which Madonna became an astronaut captures the truth conditions of the sentence *Madonna became an astronaut*. This set of worlds can be identified with the proposition expressed by this sentence. The proposition expressed by a sentence is the set of worlds in which the sentence is true. In general, propositions are sets of possible worlds, and sentences express propositions in this sense. Entailment between two sentences, then, boils down to a certain kind of relationship among the propositions they express. If A entails B, then the proposition expressed by A is a subset of the proposition expressed by B: Every possible world in the former is also in the latter. (We will talk more about set-theoretic notions like subset in the next chapter, as they are very handy in semantic theorizing.)

There are several other ways that entailment can be defined (Chierchia & McConnell-Ginet, 2000). A twist on the way we have defined it is: ‘A entails B if and only if a situation describable by A must also be a situation describable by B’. For example, 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*. Another approach to the definition is in terms of information containment: A entails B if and only if the information that B conveys is contained in the information that A conveys. 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. Another definition focuses on the non-deniability of the entailed sentence: ‘A entails B if and only if A and it is not the case that B is contradictory (can't be true in any situation). For example, *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. This last definition is connected to the primary diagnostic
test for entailments, the *defeasibility test*, which we will turn to next.

### 1.1.1 Entailment vs. implicature

Recall that example (1) implies that Hugh Laurie does not speak Cantonese. But would you say that the sentence is *false* if he *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 Hugh Laurie speaks. If he did speak Cantonese, the author of the webpage would be “lying by omission”. This implication is thus an example of a *conversational implicature*. Conversational implicatures are inferences that the hearer can derive using the assumption that the speaker is adhering to certain norms of conversation (Grice, 1975). Among these norms is the Maxim of Quantity, which requires that speakers provide as much information as needed for the information exchange. If we're on the subject of what languages Hugh Laurie speaks, and he speaks Cantonese, then the Maxim of Quantity dictates that this fact be mentioned.

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

**Jess**: So you're saying she not that attractive?
**Harry**: No, I told you she is attractive.
**Jess**: But you also said she had a good personality.
**Harry**: She does have a good personality.
**Jess**: When someone's not that attractive, they're always described as having a good personality.
**Harry**: Look, if you would ask me, “What does she look like?” and I said, “She has a good personality.” That means she's not attractive. But just because I happened to mention that she has a good personality, she could be either. She could be attractive with
a good personality, or not attractive with a good personality.

Here, Harry is pointing out that the conversational implicature from *She has a good personality* to *She is not attractive* depends on what the question under discussion is. If the question under discussion is what she looks like, then the implicature arises, but otherwise, there is no such implicature.

**Exercise 2.** What is the difference between an *implication* and an *implicature*? Explain using definitions and examples.

Conversational implicatures differ from entailments in that what is implied is not logically implied. So the implication can be cancelled without producing a contradiction. For example, one could say:

(7) Hugh Laurie speaks English, French and Spanish. In fact, he speaks Cantonese as well.

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

In contrast, entailments are not defeasible. Consider:

(8) #Hugh Laurie speaks English, French and Spanish, but he doesn't speak more than two languages.
Introduction

(The hash-mark # here indicates that the sentence is somehow odd in its interpretation. This symbol is the semantics/pragmatics equivalent of the asterisk [*] used in the syntax literature to indicate that a sentence is ungrammatical.) If your friend uttered these two sentences in succession, she would be open to the accusation that she was contradicting herself, because *Hugh Laurie speaks more than two languages* is entailed by the first sentence. In general, if you know that sentence A implies sentence B but you don't know whether this implication is an entailment or an implication, you can run this defeasibility test by constructing an example in which the A sentence is followed by the negation of the B sentence, and observing whether the result is contradictory. If so, then you have an entailment on your hands.

More specifically, to run the defeasibility test for an implication from sentence A to sentence B, the first step is to construct a text of the form A & not-B, where not-B is the negation of B, and & is whatever linking expression (but, and, and in fact, etc.) makes most sense. Constructing the negation of a sentence can sometimes be a bit tricky. It's not always a matter of just adding a not. For example, let's consider whether (9a) entails (9b).

(9) a. Some Republicans voted ‘yes’.
   b. Not all Republicans voted ‘yes’.

To run the defeasibility test on this example, we have to construct the negation of (9b). What is the negation of (9b)? Rather than adding a not, as we did with the Hugh Laurie case, we have to take one away: *All Republicans voted ‘yes’*. Then we can ask whether A & not-B is contradictory:

(10) Some Republicans voted ‘yes’; in fact all of them did.

This example is not contradictory, so the implication from (9a) to (9b) is not an entailment; it's a conversational implicature.

Now, consider the following sentence:
Everybody likes chocolate.

Which of the following is the negation of this sentence?

a. Nobody likes chocolate.
   b. Not everybody likes chocolate.

Answer: The negation of (11) is (12b): Not everybody likes chocolate. Why? How can one decide, in general? If one is unsure what the negation of a given sentence is, what method can be used to decide? When you are unsure, the definition of negation may become useful:

A sentence A is the negation of a sentence B if and only if:

A and B cannot be true at the same time, and they cannot be false at the same time.

Consider Everybody likes chocolate and Nobody likes chocolate. These two sentences cannot be true at the same time, but they can be false at the same time. Maybe some people like chocolate and some people don't. Then Everybody likes chocolate is false, and so is Nobody likes chocolate. These two sentences stand in so-called contrary opposition (can't be true at the same time, but can be false at the same time), rather than contradictory opposition (can't be true at the same time, and can't be false at the same time). The sentence that stands in contradictory opposition to Everybody likes chocolate is Not everybody likes chocolate. If the one is true, then the other must be false, and vice versa.

The negation of a sentence stands in contradictory opposition to it.

For another example to illustrate contrary vs. contradictory opposition, consider the following sentence:

The distinction between contrary and contradictory opposition plays a role in Aristotle's 'square of opposition', a fascinating idea that the reader is encouraged to seek out more information about.
The sentence *Jo is not tall* is the negation. This sentence stands in contradictory opposition to (14) If one is true, then the other must be false, and vice versa. But if we replace *tall* with its antonym, *short*, producing *Jo is short*, then the result stands in contrary opposition to (14). The two sentences can’t both be true, because one cannot be both tall and short at the same time, but they could both be false: Jo might be neither tall nor short, just somewhere in that grey zone, the ‘zone of indifference’ as Sapir (1944) called it (see e.g. Kennedy & McNally 2005).

Again, the negation of a sentence stands in contradictory, rather than contrary opposition to it, so to check whether you have constructed the negation correctly, you can check what type of opposition your sentences stand in to each other. But if you are faced with a really tricky case and you feel quite unsure what the negation is, you can always just add *It is not the case that* to the beginning of the sentence; *It is not the case that* *S* is more or less guaranteed to be the negation of *S*. For example, *It is not the case that everyone likes to eat* is a negated version of *Everyone likes to eat*. Notice that this sentence is equivalent to *Not everyone likes to eat*; when one is true, the other is true too, and when one is false, the other is false too. Thus, even if you don’t use the *It is not the case that* strategy, another way to check whether you have constructed the negation properly is to ask yourself whether your sentence is equivalent to the *It is not the case that* version.

Once you have constructed the negation, it is important that you pose the right question about the example of the form *A & not-B* to a native speaker of the language (possibly yourself, if you are a native speaker of the language). The question is not ‘Is this example grammatical?’ or ‘Is this example true?’ but rather ‘Is this example self-contradictory?’ If the answer is yes, then the *A* sentence cannot be true while the *B* sentence is false, so the implication from *A* to *B* is an entailment, rather than an implicature.
Exercise 3. For each of the following sentences, give a sentence that stands in contradictory opposition to it. Be careful to avoid giving a sentence that stands in contrary, rather than contradictory opposition.

(a) Richard is old.
(b) I danced.
(c) Everyone danced.
(d) Somebody sneezed.
(e) I danced with everyone.
(f) Sometimes I dance.
(g) I always dance.
(h) I never dance.
(i) John weighs more than Maria.
(j) Nobody weighs more than John.

Exercise 4. Samuel Bronston was a movie producer who filed for bankruptcy in 1964 after his very expensive epic film *The Fall of the Roman Empire* failed at the box office. In 1966, he was questioned under oath by his creditors regarding his overseas assets, and the exchange went as follows:

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

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

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

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

Another test that can be used to distinguish between entailments and implicatures is called the reinforcement test. The idea behind it is that if A entails B, then saying B after one has just said A sounds redundant, because B was just directly implied. Consider the following contrast:

(15) #He speaks French, German and Spanish, and he speaks more than two languages.

(16) He speaks French, German and Spanish, but he doesn’t speak Cantonese.

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

Another difference between entailments and implicatures is that implicatures tend to disappear under negation. For example, compare the following two examples:

(17) Some of the students passed the test.
(18) It's not true that some of the students passed the test.

While (17) implies that not all of the students passed the test, what's negated in (18) is not 'some but not all'.

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

**Exercise 5.** Consider the following pairs of sentences:

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

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

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

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

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

(24) a. It’s warm.
    b. It’s not hot.

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

1.1.2 Entailment vs. presupposition

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

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

while its negation is true:

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

But when presupposition comes into the mix, it can happen that neither a sentence nor its negation is true. Consider the following sentence:

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

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

(28) The theremin duo that Mozart wrote is not very famous.
The culprit behind this odd state of affairs is a presupposition accompanying the definite description *the theremin duo that Mozart wrote*. By use of this definite description, a speaker becomes committed to the existence of theremin duos written by Mozart, whether the sentence that the definite description is embedded in is positive or negative. Both (27) and (28) entail (29). Since both sentences entail something false, neither sentence is true.

(29) Mozart wrote a theremin duo.

The type of implication involved here is not an ordinary entailment, but it is not a conversational implicature either; it is a **presupposition**. To presuppose something is to take it for granted, to treat it as uncontroversial and known to everyone participating in the conversation.

The definite article *the* triggers a presupposition of existence; in this sense, it is a **presupposition trigger**. Another example of a presupposition trigger is the adverb *still*. For example, if I said (30), I would signal (31) through a presupposition.

(30) Hugh Laurie still speaks French.
(31) Hugh Laurie spoke French in the past.

Although it is not an ordinary entailment, the relation between these sentences is arguably some form of entailment; in every situation where *Hugh Laurie no longer speaks French* is true, *Hugh Laurie spoke French at some time in the past* is also true. This can also be shown using the defeasibility test. But presuppositions differ from ordinary entailments, as you can see from what happens when they are negated or put in the form of a question. If I were to negate (30) as follows (slightly awkward, but entirely comprehensible):

(32) Hugh Laurie doesn't still speak French.

then I would still be signalling that Hugh Laurie spoke French in
the past.

In fact, merely *supposing* that Hugh Laurie still speaks French also yields the implication that he spoke French in the past.

(33) If Hugh Laurie still speaks French, then he might enjoy this poem.

Here we have placed *Hugh Laurie still speaks French* in the **ANTECEDENT** position of a **CONDITIONAL** statement. Normally, material that is in the antecedent (the ‘if’ part) of a conditional is not implied. For example, the following sentence does not imply that Hugh Laurie speaks French:

(34) If Hugh Laurie speaks French, then he might enjoy this poem.

The antecedent of a conditional is for ideas that are merely entertained for the purpose of exploring a hypothetical possibility; the speaker normally does not commit herself to the material here. But presupposed information still ‘pops out’ from the antecedent of a conditional, as it were. In other words, the presupposition **PROJECTS** from the antecedent of the conditional.

We see this with questions as well. If someone were to ask,

(35) Does Hugh Laurie speak French?

They would not be implying that Hugh Laurie spoke French, of course. And yet:

(36) Does Hugh Laurie still speak French?

does imply that Hugh Laurie spoke French at some time in the past. The presupposition projects out of the yes/no question.

In general, presuppositions can be distinguished from entailments using this **PROJECTION TEST**, which assesses whether the inference in question ‘projects’ over negation, from the antecedent of a conditional statement or over question-formation. What these
environments have in common is that they are ENTAILMENT-CANCELLING environments; environments where entailments normally go to die. But presuppositions thrive in these environments. To test whether an inference from A to B is an ordinary entailment or a presupposition, one embeds A in an entailment-cancelling environment, and observes whether the B sentence is still implied. If so, then the inference projects, and is therefore behaving as a presupposition.

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, although presuppositions will be addressed as well. Implicatures will largely be left out of the discussion here.

**Exercise 6.** Use the projection test to determine whether the following implications are entailments or presuppositions. Explain how the test supports your conclusion.

(a) The flying saucer came again.
   The flying saucer has come sometime in the past.

(b) The flying saucer came yesterday.
   The flying saucer has come sometime in the past.
Exercise 7. Consider the following two sentences.

(37) a. John succeeded in losing weight.
    b. John failed at losing weight.

Intuitively, both sentences imply that John tried to lose weight \((38a)\), but the *succeed* sentence implies that he did \((38b)\) and the *fail* sentence implies that he did not \((38c)\)

(38) a. John tried to lose weight.
    b. John lost weight.
    c. John didn’t lose weight.

So there are four implications under consideration:

- \((37a)\) ‘succeed’ → \((38a)\) ‘try’
- \((37b)\) ‘fail’ → \((38a)\) ‘try’
- \((37a)\) ‘succeed’ → \((38b)\) ‘did’
- \((37b)\) ‘fail’ → \((38c)\) ‘didn’t’

For each of these in turn, determine whether it is an implicature, an ordinary entailment, or a presupposition. First, determine whether it is an implicature or an entailment (ordinary or presupposition) using the defeasibility and reinforcement tests, and then, if it is an entailment, determine whether it is an ordinary entailment or a presupposition using projection from negation, the antecedent of a conditional, and a yes/no question.

Be sure to include all of the relevant examples, observations, and reasoning in your answer, and summarize your findings by saying in general what is entailed, presupposed, and implicated (if anything), by a sentence of the form *X succeeded in Y*, and do the same for *X failed at Y*. 
1.1.3 Entailment and validity

The notion of entailment has its roots in the study of logic, which can be seen as the study of what constitutes a good argument. You have probably heard of Aristotle's syllogisms. 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:

(39) All men are mortal. (Premise 1)
    Socrates is a man. (Premise 2)
    Therefore, Socrates is mortal.

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 that is valid and whose premises are furthermore true. For example, lemonade is made from watermelon. Watermelon is a type of fruit. Therefore, lemonade is made from fruit is a valid argument (and its conclusion happens to be true!) but it is not a sound argument, because one of its premises is false.

Exercise 8. Which, if any, of the following are valid? Which, if any, 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) No mammal lacks a heart; Malala Yousafzai is a mammal; therefore, Malala Yousafzai has a heart.
(d) If \( 2 + 2 = 5 \) then the moon is made of green cheese; \( 2 + 2 = 5 \); therefore, the moon is made of green cheese.

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

**Exercise 9.** Can a valid argument have...

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

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

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

(40) Baseball is better than soccer.  
    Tennis is better than baseball.  
    Therefore, tennis is better than soccer.

but the following is not:\(^3\)

(41) Nothing is better than a long and prosperous life.  
    A ham sandwich is better than nothing.

\(^3\)Example from Dowty et al. (1981), pp. 2-3.
Therefore, a ham sandwich is better than a long and prosperous life.

While the premises and the conclusion may all be true, it seems clear that the logic is flawed. Through the study of semantics, you will develop tools for articulating exactly where it goes wrong.

1.2 Framework

The style of semantics used in this book is:

• model-theoretic,
• indirect,
• and compositional.

Let us briefly describe what we mean by these things.

1.2.1 Model-theoretic semantics

Again, knowing the meaning of a sentence does not require you to know whether the sentence is in fact true; it only requires that you be able to discriminate among situations where it is true and situations where it is false. If you can do this, then you have a grasp on the truth conditions for the sentence. The approach to semantic theory that we will pursue here is truth-conditional insofar as we will define a systematic association between sentences and their truth conditions. (By no means is this all there is to meaning, but we do take truth conditions to be one important aspect of meaning.)

Truth can be conceptualized as a relation between a sentence and a possible world. The actual world that we live in is one possible world. For instance, the pop singer Madonna is not an astronaut in the actual world, although perhaps she might have been, and we can imagine an alternative possible world in which this
proposition is true. The set of possible worlds where Madonna is an astronaut—the proposition expressed by that sentence—is a way of characterizing its truth conditions.

But we will speak primarily of truth in a model, rather than truth in a world. To a first approximation, a model is a simplified representation of a state of affairs, consisting of a set of entities (called a domain), and a specification of their properties (e.g. female, cat, green, etc.), and the relations that hold among them (e.g. mother, loves, etc.). This technique, borrowed from the tradition of mathematical logic, allows us to see clearly when one sentence entails another. Models will typically be less fine-grained than possible worlds, because the predicates and relations that it represents will typically not exhaust the full set of properties and relations that can distinguish one possible world from another. A model is like a simplified, cartoon version of a world that is easier to work with.

If a sentence is true relative to a given model, we say that the model verifies the sentence. The truth conditions for a sentence, then, can be characterized as the set of models that verify the sentence. The notion of entailment can be characterized in terms of models as well: Sentence A entails sentence B if and only if every model that verifies A also verifies B. Let us refer to this henceforth as the model-based definition of entailment.

Now, a model might be very different indeed from the actual world, so different that we might not even want to consider it possible. Is it possible that, in another possible world, you were a fried egg? In other words, if you were a fried egg, would you still be you? You might want to say that you have certain essential properties that you retain in every possible world. According to Yablo (1987, 297), the essence of a thing is “an assortment of properties in virtue of which it is the entity in question,” and “a measure of what is required for it to be that thing.” Maybe there are some possible worlds where you are blonde, and other worlds where you are not, but no possible worlds where you are a fried egg, because
fried eggs lack some of your essential properties. Likewise, there might be some worlds where water is poisonous for humans, but a theorist might well insist that there are no worlds where water is composed solely of nitrogen, since water is by its essential nature composed of hydrogen and oxygen. If we allow water to be composed solely of nitrogen in some models, then ‘X is water’ does not entail ‘X contains hydrogen’ under the model-based definition of entailment (A entails B iff every model that verifies A also verifies B). For another example, ‘John is a bachelor’ intuitively entails ‘John is unmarried’, because whenever the former is true, the latter is true too. But models are in principle blind to the inherent relations among the content words in a language, so there might be a model in which the property ‘bachelor’ is applied to some married people. If so, then ‘John is a bachelor’ does not entail ‘John is unmarried’ according to the model-based definition of entailment.

When models are taken to be so totally unconstrained, the model-based definition of entailment characterizes logical consequence in the sense of Tarski (1983 [originally published 1936]). Tarski wished to distinguish a type of consequence relation that is purely logical, and totally independent of empirical knowledge, i.e., knowledge about the world. This notion of logical consequence is very strict. It only captures entailment relations that hold regardless of who has what properties and who bears what relations to who else, like the fact that ‘Mary drinks and Bob smokes’ logically implies ‘Mary drinks’. Only the meanings of words like ‘and’, ‘or’ and ‘if’ are universal across all models, so what we end up with is just a theory of those words. (These words, like ‘and’, ‘or’ and ‘if’, correspond to the logical constants of the formal representation language, as opposed to the non-logical constants like happy. We will return to this distinction after introducing logic.)

Suppose we instead characterize entailment in terms of possible words instead of models: A entails B iff in every possible world where A is true, B is true too. Then ‘John is a bachelor’ does entail
‘John is married’, given that it is impossible to be a married bachelor. We might call this relation NECESSARY CONSEQUENCE, in order to distinguish it from LOGICAL CONSEQUENCE.

Both logical and necessary consequence can be captured, or at least approximated, in a model-theoretic framework. The difference corresponds to whether or not we constrain our models so that they only describe possible worlds. We can do so using something that Montague called MEANING POSTULATES. Meaning postulates restrict the set of models so that impossible ones are ruled out. For example, we might have a meaning postulate saying that there are no married bachelors in any model. As a result, ‘John is a bachelor’ will entail ‘John is unmarried’ even under the model-based definition of entailment: If there are no models where bachelors are married, then every model where ‘John is a bachelor’ is true will be a model where ‘John is unmarried’ is true as well. Generally, if the space of models is constrained via meaning postulates so that every model is possible, then our model-based characterization of entailment (in every model where A is true, B is true too) corresponds to necessary consequence. Following Montague, this is the strategy we will aim for, although we will not be very diligent about spelling out meaning postulates.

1.2.2 Indirect interpretation

Rather than associating natural language expressions directly with models, we will characterize this relationship indirectly, in two steps. (We use EXPRESSION to describe any meaningful string in a language, such as a sentence or a phrase.) Mediating between natural language expressions and models will be a logical REPRESENTATION LANGUAGE whose expressions are very straightforwardly evaluated with respect to a model, because they are unambiguous and their meaning does not change depending on context. Sentences of natural languages will first be translated into this formal representation language. These representation language expressions are then associated with models. The truth conditions
for the formal language translations are inherited by the corresponding natural language sentences. This method is known as indirect interpretation, and it is the method used in Richard Montague's famous paper, ‘The Proper Treatment of Quantification in Ordinary English’. Another way to go about things would be to skip the representation 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, and it is found in the Heim & Kratzer (1998) textbook.

For example, the English name *Bart* will be translated into our representation language as *bart*, which in turn denotes the individual Bart Simpson.

<table>
<thead>
<tr>
<th>English Logic Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Bart</em> <em>bart</em> Bart Simpson</td>
</tr>
</tbody>
</table>

We refer to Bart Simpson in the meta-language using his name, "Bart Simpson". (We were tempted to use a pictorial representation of Bart Simpson in the third column in the table above, in order to drive home the point that it is really the individual himself that we mean to indicate, but that would unfortunately constitute a violation of copyright law!) In the system that we will set up, the English name *Bart* indirectly becomes associated with this meaning, by virtue of the fact that it is associated with the symbol *bart*, which is part of the representation language. This representation language symbol *bart* will be directly associated with the individual Bart Simpson via the semantic rules for first-order logic.

Although it is only a stylistic choice, indirect interpretation offers a number of practical technical advantages over direct interpretation. The main advantage derives from the fact that natural language is ambiguous and context-sensitive and vague, while our logical representation language is not. Having a non-vague, non-context-sensitive, non-ambiguous representation language greatly simplifies the task of assigning meanings to expressions. Moreover, since our representation language is well-defined, it is
possible to know with certainty when one expression of the representation language is equivalent to another; for example, we can understand when and why the $\beta$-reduction principle of lambda calculus can be relied upon. This is important, because we use $\beta$-reduction extremely frequently when we derive meanings compositionally for sentences. A more practical advantage is that it allows our meaning representations to be more concise, so they can fit on a tree diagram showing the compositional derivation of the meaning of a sentence. Finally, and importantly, it also meshes well with the Lambda Calculator, pedagogical software that can be used in conjunction with this book.

1.2.3 Compositional semantics

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. How? Gottlob Frege conjectured that meanings always combine through saturation: the meaning of one expression is unsaturated, and it is saturated by the meaning of the expression it combines with. Here is how he puts it:  

It is astonishing what language accomplishes. With a few syllables it expresses a countless number of thoughts, and even for a thought grasped for the first time by a human it provides a clothing in which it can be recognized by another to whom it is entirely new. This would not be possible if we could not distinguish parts in the thought that correspond to parts of the sentence, so that the construction of the sentence can be taken to mirror the construction of the thought ... If we thus view thoughts as composed of simple parts and take these, in turn, to correspond to simple sentence-parts, we can understand how a few sentence-parts

---

can go to make up a great multitude of sentences to which, in turn, there correspond a great multitude of thoughts. The question now arises how the construction of the thought proceeds, and by what means the parts are put together so that the whole is something more than the isolated parts. In my essay “Negation,” I considered the case of a thought that appears to be composed of one part which is in need of completion or, as one might say, unsaturated, and whose linguistic correlate is the negative particle, and another part which is a thought. We cannot negate without negating something, and this something is a thought. Because this thought saturates the unsaturated part or, as one might say, completes what is in need of completion, the whole hangs together. And it is a natural conjecture that logical combination of parts into a whole is always a matter of saturating something unsaturated.

As we will discuss further in the next chapter, Frege proposed to model this “saturation” using mathematical functions. The composition rule that implements this intuition is called Function Application. Following Heim & Kratzer (1998), we will supplement this very general rule with two more ways of putting the meanings of complex expressions together (Predicate Modification and Predicate Abstraction). But that’s it; our theory of semantic composition is quite slim and trim, although it handles a wide range of phenomena.

**Final word**

Consider this book a starter kit for a theory of semantics. If you understand the foundations well, you will be able to modify it to suit your purposes. Accounting for a wider range of phenomena may well require foundational changes to the theory, but if you
understand the foundations of this simple theory well enough to go in and adjust them, then you will be in a position to make a fundamental contribution to the theory of natural language semantics.

**Exercise 10.** What is *truth-conditional* semantics?

**Exercise 11.** What is *model-theoretic* semantics?

**Exercise 12.** What is the difference between direct and indirect interpretation? Which style will be used in this book?

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

**Exercise 14.** What is Frege’s conjecture?  
(Hint: It has to do with the concept of a function.)
2 | Sets, Relations, and Functions

2.1 Introduction

In the previous chapter, we invoked the concept of subset in our characterization of entailment. If a sentence is taken to express a proposition, construed as a set of possible worlds, then the notion of entailment can be characterized in terms of subset as follows: A entails B if and only if the proposition expressed by A is a subset of the proposition expressed by B. (This boils down to our original definition: Whenever A is true, B is true too.) Characterizing entailment is only one of the many uses for set-theoretic concepts in semantic theorizing; there are many more.

This chapter provides a brief introduction to set theory, including relations between sets like subset and superset, as well as operations on set like intersection, union and complement. We will also use sets to characterize relations and functions. Functions play a particularly important role in semantic theorizing, as they give us a way of making precise Frege's conjecture that composition is about saturation.

2.2 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 write:

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

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

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

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

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 **singleton**. A set can even be empty. The set with no elements at all is called the **empty set**, written either like this:

\[
\{\}\n\]
or like this:

\[
\emptyset
\]

The **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.** What is the cardinality of the following sets?

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

(b) \( \emptyset \)
In the kind of set theory that linguists typically use, elements may be either concrete (like the beige 1992 Toyota Corolla the first author sold in 2008, you, or your computer) or abstract (like the number 2, the English phoneme /p/, or the set of all 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:

\[
\{a, \{b\}\} \not\subseteq \{a, b, c\}
\]

(The slash across the \( \subseteq \) symbol negates it, so \( \not\subseteq \) 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. (It depends whether b is a set; we have made no assumptions about what b is.) The following is a true statement, though:

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

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

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

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

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 ⊂ B iff (i) for all x: if \( x \in A \) then \( x \in B \) and (ii) \( A \neq B \).

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

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 ia member of \( A \).

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

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

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

Subset and superset are relations between sets, which either hold or fail to hold. Other elements of the set theoretic vocabulary express operations on sets, transforming one or more sets into a new set. The principal operations on sets include intersection, union, and complement.

The intersection of \( A \) and \( B \), written \( A \cap B \), is the set of all entities \( x \) 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 and. For example, if someone tells you that John is a lawyer and a doctor, then you know that John is in the intersection between the set of lawyers and the set of doctors. If the dashed circle in the following diagram represents
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.

The English determiner *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, *some musician snores* should be true if there is some individual which is both a musician and a snorer.

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

---

**Exercise 2.** Here are three Venn diagrams:

(a) ![Venn Diagram A](image)

(b) ![Venn Diagram B](image)
And here are three statements in set-theoretic language:

1. \( A \cap B = \emptyset \)
2. \( A \cap B \neq \emptyset \)
3. \( A \subseteq B \)

For each of the Venn diagrams, say (i) which of the three set-theoretic statements it matches, and (ii) which of the following three quantifiers it best represents: *some*, *every*, or *no*.

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

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

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

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

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

We can also talk about *subtracting* one set from another. The **difference** of $A$ and $B$, written $A - B$ or $A \setminus B$, is the set of all 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. This can be written $\bar{A}$.

**Exercises on sets**

The following exercises are taken from Partee, ter Meulen and Wall, *Mathematical Methods in Linguistics*. 
Exercise 4. 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) \( c \in A \) \quad (g) \( D \subseteq A \) \quad (m) \( B \subseteq G \)
(b) \( c \in F \) \quad (h) \( A \subseteq C \) \quad (n) \( \{ B \} \subseteq G \)
(c) \( c \in E \) \quad (i) \( D \subseteq E \) \quad (o) \( D \subseteq G \)
(d) \( \{ c \} \in E \) \quad (j) \( F \subseteq A \) \quad (p) \( \{ D \} \subseteq G \)
(e) \( \{ c \} \in C \) \quad (k) \( E \subseteq F \) \quad (q) \( G \subseteq A \)
(f) \( B \subseteq A \) \quad (l) \( B \in G \) \quad (r) \( \{ \{ c \} \} \subseteq E \)

Exercise 5. 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 6. 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) $A \cap B$

(m) $B - A$
(n) $C - D$
(o) $E - F$
(p) $F - A$
(q) $G - B$

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

(a) $A \cup B$
(b) $A \cap B$
(c) $A \cup (B \cap C)$
(d) $C \cup A$
(e) $B \cup \emptyset$
(f) $A \cap (B \cap C)$
(g) $A - B$

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

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

### 2.3 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.

#### 2.3.1 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. Like the elements of 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:

\[(3, \{10, 12\})\]

In this ordered pair, the first member is the number 3, and the second member is the ordered pair \((10, 12)\). Note that \((3, \{10, 12\})\) is not the same thing as \((3, \{10, 12\})\). 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* \((10, 12)\).

**Exercise 8.** True or false?

(a) \(\{3, 3\} = \{3\}\)
(b) \(\{3, 4\} = \{4, 3\}\)
(c) \((3, 4) = (4, 3)\)
(d) \((3, 3) = (3, 3)\)
(e) \(\{(3, 3)\} = (3, 3)\)
(f) \(\{(3, 3), (3, 4)\} = \{(3, 4), (3, 3)\}\)
(g) \((3, \{3, 4\}) = (3, \{4, 3\})\)
(h) \(\{3, \{3, 4\}\} = \{3, \{4, 3\}\}\)

### 2.3.2 Relations

As mentioned above, transitive verbs like *love*, *admire*, and *respect* are sometimes thought of as *relations* between two individuals. The ‘love’ relation corresponds to the set of ordered pairs of individuals such that the first member loves the second member. Sup-
pose John loves Mary. Then the pair \( \langle \text{John}, \text{Mary} \rangle \) is a member of this relation.\(^1\)

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

A preposition like \textit{in} denotes a relation between \textit{two} individuals; that is, it denotes a \textit{binary relation}. The preposition \textit{between}, on the other hand, expresses a \textit{ternary relation}, i.e., a relation between three objects (\( a \) is between \( b \) and \( c \)). A ternary relation can be modelled as a set of ordered triples. For example, the ternary relation denoted by \textit{between} contains the following triples:

\[
\langle \text{Alabama}, \text{Mississippi}, \text{Georgia} \rangle  \\
\langle \text{Togo}, \text{Ghana}, \text{Benin} \rangle
\]

as Alabama is between Mississippi and Georgia and Togo is between Ghana and Benin. A \textit{quaternary} relation corresponds to a set of ordered 4-tuples. The verb \textit{sell}, for example, could be modelled as a quaternary relation between a seller, a buyer, an object, and a price.

A binary relation is associated with a \textsc{domain} and a \textsc{codomain}. The \textsc{domain} is a set from which the first element of any ordered pair is drawn, and the \textsc{codomain} is a set from which the second

\(^1\)The set of ordered pairs such that the first member loves the second is called the \textit{extension} of the ‘love’ relation. Sometimes a relation and its extension are identified; on this view, a relation just \textit{is} a set of ordered pairs. Sometimes a relation and its extension are kept conceptually separate; on such a view, a relation is some kind of abstract entity, such as an algorithm for determining whether two people love each other, and the extension is determined by it but not the same thing. We will tend to blur the distinction between a relation and its extension for the purpose of cutting down on the length of our sentences.
element of any ordered pair is drawn. The set of things that are actually found in some ordered pair in the relation might be a strict subset of the domain. If every element of the domain stands as the first member of an ordered pair in the relation, then the relation is said to be total. For example, we might define a ‘girlfriend’ relation whose domain is the set of human beings and whose range is the set of female individuals. Since not every human being has a girlfriend, this would not be a total relation.

Similarly, we set of things that actually stand as the second member of a pair in the relation may not exhaust the entire codomain. The set of things that actually stand as the second member of a pair in the relation is called the range. The range is always a subset of the codomain, possibly a proper subset.

Depending on the type of relation, different inferences may be licensed. A symmetric relation is one that licenses inferences like this:

\(1\) Paul is standing next to George.
Therefore, George is standing next to Paul.

In general, a relation is symmetric if and only if: For any \(a\) and \(b\) if \((a, b)\) is in the relation, then \((b, a)\) is also in the relation. Another example of a symmetric relation is marriage: If \(a\) is married to \(b\), then \(b\) is also married to \(a\).

A transitive relation is one that licenses inferences like this:

\(2\) Paul is taller than George.
George is taller than Ringo.
Therefore, John is taller than Ringo.

In general, a relation is transitive if and only if: For any \(a, b,\) and \(c\), if \((a, b)\) and \((b, c)\) are in the relation, then \((a, c)\) is also in the relation. Another example of a transitive relation would be ‘before’: If \(a\) is before \(b\), and \(b\) is before \(c\), then \(a\) is before \(c\).
Exercise 9. One of the following arguments is valid, and the other is not.

(3) The singer is the drummer's brother.
    Therefore, the drummer is the singer's brother.

(4) The singer is the drummer's sibling.
    Therefore, the drummer is the singer's sibling.

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

Exercise 10. One of the following arguments is valid, and the other is not.

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

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

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

Exercise 11. ABBA is composed of two couples: Björn and Agneta, and Frida and Benny. The ‘partner’ relation can be expressed as
the following set of pairs:

\{ \{\text{Agneta, Björn}\}, \{\text{Björn, Agneta}\}, \{\text{Frida, Benny}\}, \{\text{Benny, Frida}\}\}

(a) Is the ‘partner’ relation symmetric? Explain why or why not.
(b) Is the ‘partner’ relation transitive? Explain why or why not.

2.4 Functions

Recall that Frege said, “it is a natural conjecture that logical combination of parts into a whole is always a matter of saturating something unsaturated.” What does it mean to “saturate something unsaturated”? According to Frege, this involves the concept of a function. Consider the following quotation from Frege:\footnote{Frege, “Function and Concept” (1891), trans. in M. Black and P. Geach, 
Translations from the Philosophical Writings of Gottlob Frege (Oxford, Basil Blackwell, 1960), pp. 21-41, at p. 31.}

Statements in general, just like equations or inequalities or expressions in Analysis, can be imagined to be split up into two parts; one complete in itself, and the other in need of supplementation, or “unsaturated.” Thus, for example, we split up the sentence “Caesar conquered Gaul” into “Caesar” and “conquered Gaul.” The second part is “unsaturated” - it contains an empty place; only when this place is filled up with a proper name, or with an expression that replaces a proper name, does a complete sense appear. Here too I give the name “function” to what this “unsaturated” part stands for. In this case the argument is Caesar.

Thus *conquered Gaul*, for Frege, expresses a function, which takes *Caesar* as an argument.
Exercise 12. How does Frege's Conjecture relate to the concept of mathematical function?

The word ‘function’ has many meanings, but here we are using it in its mathematical sense. You can think of a mathematical function like a vending machine: It takes an INPUT (e.g. a specification of which item you would like to buy), and gives an OUTPUT (e.g. a particular bag of chocolate-covered raisins). (Let us set aside the fact that vending machines typically also require an input of money; this constitutes an additional input.) An example of a function is something that takes a word and gives back its length in letters. For example, given princess or football it would give back 8. Given strawberry it would give back 10.

Technically, functions are a special type of relation, so a function is a set of ordered pairs. The following ordered pairs would be in the first function just described:

\[(\text{princess}, 8)\]
\[(\text{football}, 8)\]
\[(\text{strawberry}, 10)\]

But not every relation is a function. A relation from \(A\) to \(B\) is a function only if every element of \(A\) is mapped to one and only one member of \(B\). In this case, we have a relation from words to integers. Two different elements of \(A\) may map to the same member of \(B\), but for every element of \(A\), there can be only one member of \(B\) that it maps to. For example, both princess and football are mapped to the number 8 by this ‘length in characters’ function, but the only value that princess is mapped to is 8.

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 out-
Sets, Relations, and Functions

In Figure 2.1, the relations depicted are not functions. In Figure 2.2, the relations depicted are functions.

Formally, a function is a special kind of relation. (Very special! Functions lie at the heart of modern formal semantics.) Here is the official definition of a function. 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 $A$.

(Here by ‘domain’ of $R$ we mean the set of entities that appear in the first place in some ordered pair in the relation.)

For example, $\{\langle 0, 0 \rangle, \langle 0, 1 \rangle, \langle 1, 1 \rangle\}$ cannot be a function because 0 is mapped to two different ‘outputs’: 0 and 1. Here is a visual representation of that relation, which might make it easier to see:
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 13. Some nouns in English express relations; these are called *relational nouns*. A special class of relational nouns expresses functions; these are sometimes called *functional nouns*. Which of the following might be called functional nouns? When answering this question, assume domains where the relations in question are defined. For example, when deciding whether *height* is a function, assume a domain that only contains objects that can have height to begin with.

(a) *height*
(b) *center*
(c) *edge*
(d) *part*
(e) *age*
(f) *citizenship*

**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”):

\[
\{(\text{Homer}, 0), (\text{Marge}, 0), (\text{Bart}, 1), (\text{Lisa}, 1), (\text{Maggie}, 1)\}
\]
Here is an alternative way of writing the same thing:

\[
\begin{array}{c|c}
\text{Homer} & 0 \\
\text{Marge} & 0 \\
\text{Bart} & 1 \\
\text{Lisa} & 1 \\
\text{Maggie} & 1 \\
\end{array}
\]

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 \textsc{characteristic function} of $S$. The function we have just been looking at is the characteristic function of the set \{Bart, Lisa, Maggie\}.

\textbf{Exercise 14.} Recall that ABBA is composed of two couples: Björn and Agneta, and Frida and Benny, and the ‘partner’ relation can be expressed as the following set of pairs:

\[
\{ (\text{Agneta, Björn}), (\text{Björn, Agneta}), (\text{Frida, Benny}), (\text{Benny, Frida}) \}
\]

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

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

(c) True or false:

\[
\text{partnerOf}(\text{partnerOf}(\text{Björn})) = \text{partnerOf}(\text{Agneta})
\]
Exercise 15. Recall that the characteristic function of a set is a function that maps every member of that set to 1, and every non-member (in some specified larger set) to 0. For example, the characteristic function of the set of female individuals in ABBA is:

\{\langle\text{Agneta}, 1\rangle, \langle\text{Björn}, 0\rangle, \langle\text{Frida}, 1\rangle, \langle\text{Benny}, 0\rangle\}\n
(a) Give the characteristic function of the set of male individuals in ABBA.

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

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

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

2.5 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:
(7)  
   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).

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

who had *ever* read *anything* about

\[
\begin{align*}
\text{never} \\
\text{rarely} \\
\text{seldom} \\
\text{*usually} \\
\text{*always} \\
\text{*sometimes}
\end{align*}
\]

ever eat *anything* for breakfast *anymore*.

(8)  

(9)  

(10)  
   a. Susan finished her homework \{ without \{ *with \} \} *any* help.
   b. Susan voted \{ against \{ *for \} \} *ever* approving *any* of the proposals.

(11)  

(12)  

It’s \{ hard \{ *easy \} \} to find *anyone* who has *ever* read *any-
thing much about phrenology.

(13) John \( \{ \begin{align*}
doubt & \\
deny & \\
*believe & \\
*hope & \\
\end{align*} \) that anyone would ever discover that the money was missing.

(14) It \( \{ \begin{align*}
*is\ likely & \\
*is\ certain & \\
*is\ surprising & \\
*is\ unsurprising & \\
\end{align*} \) 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:

(15) 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:

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

This shows that the ability to license negative polarity items is not a simple yes/no matter for each lexical item.
Building on work by Fauconnier (1975), Ladusaw (1980) illustrated a correlation between NPI licensing and “direction of entailment”. A simple, positive sentence containing the word cellist will typically entail the corresponding sentence containing the word musician:

(17) 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 (5) proceeds from lower (more specific) to higher (more general), hence “upwards”.

An entailment by a sentence of the form [...] A [...] to a sentence of the form [...] B [...] where A is more specific than B can thus be labelled an UPWARD ENTAILMENT. Here is another upward entailment:

(18) a. Some cellists snore.
    b. ⇒ Some musicians snore.

But negation and the determiner no reverse the entailment pattern:

(19) a. Mary isn’t a musician.
    b. ⇒ Mary isn’t a cellist.
(20)  a. No musicians snore.  
b.  ⇒ 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.

(21)  a. No musician snores.  
      ⇒ No cellist snores. (downward)
  b. No musician snores.  
      ⇒ No musician snores loudly. (downward)

(22)  a. Every musician snores.  
      ⇒ Every cellist snores. (downward)
  b. Every musician snores loudly.  
      ⇒ Every musician snores. (upward)

(23)  a. Some cellists snore.  
      ⇒ Some musicians snore. (upward)
  b. Some musician snores loudly.  
      ⇒ Some musician snores. (upward)

The Fauconnier-Ladusaw 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.

\[ \text{S stands for “Sentence”, DP stands for “Determiner Phrase”, VP stands for “Verb Phrase”, D stands for “Determiner”, and NP stands for “Noun Phrase”; it stands for an intermediate phrase level in between nouns and noun phrases.} \]
So no licenses NPIs in the NP, and the expression no musician licenses NPIs and downward entailments in the VP. Every licenses NPIs in the NP, 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 $XY$ 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 $XY$ 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

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.
element on the right, \( Y \).) In general, a determiner \( \delta \) is left downward monotone if \( \delta XY \) entails \( \delta X'Y \) for all \( X' \) that are subsets of \( X \).

A determiner \( \delta \) is right downward monotone if \( \delta XY \) entails \( \delta XY' \) for any \( Y' \) that is a subset of \( Y \). Let us consider whether every is right downward monotone. Suppose that every \( XY \) 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:

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 17. Consider the following data:

(24) At most five \textit{[of the cities I have \textbf{ever} visited]} \textit{[have decent bike infrastructure]}.

(25) At most five \textit{[of the cities I have visited]} \textit{[have \textbf{any} decent bike infrastructure \textbf{at all}]}.

(26) \textbf{*At least five \textit{[of the cities I have \textbf{ever} visited]} \textit{[have decent bike infrastructure]}}.

(27) \textbf{*At least five \textit{[of the cities I have visited]} \textit{[have \textbf{any} decent bike infrastructure \textbf{at all}]}).

This shows that \textit{at most five} licenses Negative Polarity Items in the partitive phrase it combines with as well as the verb phrase, and \textit{at least five} licenses negative polarity items in neither position. In this section, let us consider whether the distribution of negative polarity items with these quantifiers fits The Fauconnier-Ladusaw generalization about downward entailment (that NPIs are licensed in downward-entailing environments).

In particular consider whether \textit{at most five} and \textit{at least five} produce downward-entailing environments in the partitive phrase it combines with, and in the verb phrase. You'll need to construct four pairs of examples, one pair for each of the environments under consideration.

Note: Your examples should \textbf{not} contain NPIs; your goal is just to determine whether the environment is downward-entailing.
Based on your observations, does the Fauconnier-the Fauconnier-Ladusaw generalization generalization hold up for at least and at most? Discuss.

**Exercise 18.** For each of the examples in (8), (9), and (10b), check whether the Fauconnier-Ladusaw generalization'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 (21b), (22b) and (23b)).

Note that although the Fauconnier-Ladusaw generalization'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, some authors, including Zwarts (1995) and Giannakidou (1999), have offered a theory of negative polarity item licensing based on a notion called ‘veridicality’.

Another complicating 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.

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

(29) a. John will replace the money if he gets to France.
   b. $\Rightarrow$ John will replace the money if he gets to Paris.
Whether or not one gets the intuition that (28a) entails (28b) seems to depend on whether or not one assumes that John will get to Paris. Assuming that John will get to Paris, (28a) does seem to imply (28b) because John will get to France either before he gets to Paris or simultaneously. This is the kind of observation that motivated 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.

4Similarly, 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.
3 | First-order logic

3.1 Introduction

Natural languages like English are full of ambiguities and fuzzy boundaries and implicit sensitivities to myriad dimensions of context. Because of this, it is convenient for us to associate natural language expressions with meanings in a two-step manner: First, we translate sentences in natural language to formulas in an clearly defined logic, and then we associate those logical formulas with meanings (indirect interpretation). Since the logical language is an artificial construct that the theorist can design according to whim, it can be unambiguous, with a straightforward association between expressions of the logic and their meanings. (Recall that we use the term expression to describe any meaningful string in a language, such as a sentence or a phrase.)

In order to carry out this program, it is necessary to define a formal logical language that we can associate natural language expressions with. To that end, this chapter introduces the language of first-order logic. The lambda calculus, which we will ultimately use as our representation language for meanings in nat-

1Why is it called first-order logic? Among the basic expressions of first-order logic are names, which denote individuals, and predicates, which may be either unary, like Happy (denoting a set), or binary, like Loves (denoting a relation). What makes the language first-order is that predicates only apply to individuals, and not to other predicates. Second-order logic allows predicates to apply to other predicates. We will consider a higher-order system in Chapter 4.
ural language, is presented in the next chapter.

Note: Recall the distinction between object language and meta-language: object language is the language we are theorizing about in our meta-language. In our framework, we actually have two object languages, English and the logical representation language. Thus, the representation language is an object language for us. In the next two chapters, the logic will be the main object language in question, and English (supplemented with some formal vocabulary such as various Greek letters and set-theoretic notions) will be our meta-language.

### 3.2 Propositional logic

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

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

This argument has two premises, and a conclusion. The conclusion follows logically from the collection of the premises: As long as the premises are both true, the conclusion is true too. One might quibble with the premises, but as long as they are both granted, then the conclusion follows. Hence, the argument is valid.

The argument in (1) is an instance of the argument form known as Modus Ponens. Modus Ponens describes arguments matching the following general template:

(2) If P then Q.
    P.
    Therefore, Q.
**Exercise 1.** Give another argument using Modus Ponens.

Now consider this superficially similar argument:

(3) If it rained last night then the lawn is wet. (Premise 1)
The lawn is wet. (Premise 2)
Therefore it rained last night. (Conclusion)

One might be tempted to think that this argument is valid, but it is not. The reason is that the premises might be true while the conclusion is false. It may well be true that the lawn gets wet whenever it rains, and that the lawn is wet. But if something other than rain can cause the lawn to become wet, perhaps a sprinkler, then the conclusion might still be false. Because the conclusion is not necessarily true when both of the premises are true (that is, because the conclusion is not entailed by the collection of the premises), the argument is not valid. This argument form is called **asserting the consequent**, and it is a well-known **fallacy**—an argument form that is not valid. Asserting the consequent has the following general template.

(4) If P then Q.
Q.
Therefore, P.

**Exercise 2.** Give another fallacious argument using Asserting the Consequent.

Propositional logic aims to capture the difference between valid argument forms and fallacies, where the templates involved have placeholders for entire sentences.
Exercise 3. For both of the following argument forms, say whether it is valid or a fallacy.

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

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

3.2.1 Formulas and propositional letters

Let us begin our study of propositional logic with the notion of a PROPOSITIONAL LETTER (also called propositional variable or sentential letter). A propositional letter is a symbol that represents a proposition, roughly, the kind of thing that is expressed by a simple declarative sentence that does not contain any of the words and, or, not, if, then.

The letters $P$ and $Q$ as we have used them above can be thought of as propositional letters. In this chapter, we will adopt the following inventory of propositional letters:

Syntactic Rule: Propositional letters
$P$, $Q$, and $R$ are propositional letters.

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

This is a choice we made, and various deviations from it would still be considered propositional logic. In principle, any set of
symbols can be used as propositional letters. Typical choices besides ours are $p$, $q$, $r$ and $a, b, c$. When more letters are needed, it is customary to use primes as in $P, P', P'', p, p', p''$ etc. Different choices of letters will give rise to different propositional languages, though the difference between these languages is immaterial. We will refer to the specific propositional language we are building up as $L_{\text{Prop}}$.

Now, what is the denotation of a formula? Recall from Chapter 1 that propositions can be either true or false. Building on the idea that formulas express something that can be either true or false, Frege (1892 [reprinted 1948]) suggested that the denotation of a sentence is a truth value: either ‘the True’ or ‘the False’. In Chapter 1 we have represented a proposition as the set of all those possible worlds at which it is true. In propositional logic, the notion of a possible world is formally implemented with the help of an interpretation function. An interpretation function for a given propositional language is a function that maps each propositional letter of that language to a truth value. We will write $I$ for such functions, and we will write 1 for ‘the True’ and 0 for ‘the False’. Here is an example of an interpretation function for $L_{\text{Prop}}$.

$$I_1 = \begin{cases} P & \rightarrow & 1 \\ Q & \rightarrow & 1 \\ R & \rightarrow & 0 \end{cases}$$

In the logic we define in this section, every propositional letter is a formula (sometimes also called well-formed formula or wff). Formulas are the sorts of logical expressions that can express propositions. In propositional logic, every expression is a formula and vice versa. Later on, though, we will see that in first-order logic there are also expressions, so-called terms, that are not formulas.

We generally speak of formulas being true or false “under an interpretation” (that is, given an interpretation function) or “with respect to an interpretation” or “in a possible world” or “in a model”.
Formulas in propositional logic may be built up from smaller formulas, just as natural language expressions may be built up from smaller expressions. To define and interpret formulas of arbitrary size we will lay down syntax rules (also called rules of formation) and semantic rules. Syntax rules specify how to build formulas and semantics rules specify how to interpret them, that is, how to map them to true or false. These rules will be defined recursively, in the sense that they specify the well-formedness and truth values of larger formulas in terms of smaller ones. The semantic rules will be compositional, in the sense that they will assign truth values to larger formulas in ways that depend only on the truth of the smaller formulas (rather than on their shape or length, for example). Because the connectives in propositional logic that we will encounter obey such compositional rules, they are also called truth-functional.

As we assign truth values to complex formulas in terms of smaller ones, it is useful to distinguish the base case from the inductive step. To do this, rather than extending \( I \) from propositional letters to arbitrary formulas, we will introduce a denotation function \([\cdot]^I\) (pronounced “the denotation of . . . under \( I \)”) and define it in terms of \( I \). Like \( I \), this function maps formulas to their truth values. While the domain of \( I \) will continue to be the set of propositional letters, the domain of \([\cdot]^I\) will be the set of all formulas of our propositional language.

**Notational convention**

Let \([\phi]^I\) stand for the truth value of any expression \( \phi \) with respect to interpretation function \( I \).

The following semantic rule makes clear that in the case of propositional letters, \( I \) and \([\cdot]^I\) coincide:
Semantic Rule: Propositional letters
If \( \phi \) is any propositional letter, then

\[
[\phi]^{I} = I(\phi).
\]

Here, the Greek letter \( \phi \) (“phi”) is a metavariable, a symbol which stands for a formula of propositional logic. Typical metavariables for propositional logic include \( \phi \) (sometimes written \( \varphi \)) and \( \psi \) (“psi”). Metavariables are not themselves part of logic; they are part of the metalanguage that we use to talk about logic. Our metalanguage in this chapter is English with a few extra symbols such as metavariables thrown in.

### 3.2.2 Boolean connectives

Formulas in propositional logic can be combined and assembled into larger formulas by using the so-called boolean connectives (named after the logician George Boole). These connectives correspond intuitively to the English expressions and, or, not, if . . . then, and if and only if (often abbreviated iff). The meanings of the first three of these words are intimately connected with each other. To illustrate: Suppose you ask your friend if she is free today or tomorrow and she says no. That means that she’s not free today, and she’s not free tomorrow. So in general:

\[ \text{not } [P \text{ or } Q] \]

means

\[ [\text{not } P] \text{ and } [\text{not } Q] \]

On the other hand, suppose you ask her if she’s free today and tomorrow, and she says no. That is not quite as strong; it means that either she’s not free today or she’s not free tomorrow (or both). Thus
not [P and Q]
means

[not P] or [not Q]

These are De Morgan’s Laws. By specifying a syntax and an interpretation for connectives corresponding to and, or, and not, we can capture the logical relationships between these sentences.

The term \textit{connective} is used in logic for symbols that connect formulas, or attach to them, to form new formulas. A propositional letter standing alone is called an \textit{atomic formula}, while formulas that are formed with the help of connectives are called \textit{complex formulas}. Two examples of connectives in propositional logic are the symbol \(\land\) (sometimes written &), meaning ‘and’, and the symbol \(\lor\) (sometimes written |), meaning ‘or’. Symbols such as conjunction and disjunction are called \textit{binary connectives}, because they join two formulas together. The negation symbol \(\neg\) (sometimes written ~) is called a \textit{unary connective}, because it applies to a single formula to produce a new one.

If I say, \textit{Susan does not volunteer on Monday}, then my statement is true if and only if it is not the case that Susan volunteers on Monday. This is called a \textit{negation}. Let’s express the rule using symbols to bring out its structure, using the propositional letter \(P\) to represent the proposition \textit{Susan volunteers on Monday}. We then represent \textit{Susan does not volunteer on Monday} as follows:

\[ \neg P \]

This is a formula and can be read ‘it is not the case that P’, or simply, ‘not P’. The \(\neg\) symbol represents ‘it is not the case that’. In general:

\textbf{Syntax Rule: Negation}
If \(\phi\) is a formula, then \(\neg\psi\) is also a formula. (This is called the \textit{negation} of \(\phi\).)
Now, this ¬ symbol should be interpreted in such a way that ¬P is true whenever both P is false, and vice versa. There are two possibilities to consider: P is true; P is false. The interpretation of ¬ can be represented using a truth table, as follows. Here, we use the number 1 to represent “true” and the number 0 to represent “false”.

<table>
<thead>
<tr>
<th></th>
<th>¬P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

This says: If P is true, then ¬P is false; and if P is false, then ¬P is true.

Such truth tables are quite practical for carrying out certain calculations. We will express the information contained in this truth table in a different format as our official semantic rule:

**Semantic Rule: Negation**

If φ is a formula, then [¬φ]I = 1 if [φ]I = 0, and 0 otherwise.

Let us now consider the binary connectives. If I say, Susan volunteers on Monday and Wednesday, then my statement is true if and only if it is both true that Susan volunteers on Monday and that she volunteers on Wednesday. This is called a conjunction. Let’s express the rule using symbols to bring out its structure.

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

Let us represent “I’ll see you today and tomorrow” as follows:

\[ [P \land Q] \]

This is a formula and can be read ‘P and Q’. The ∧ symbol represents ‘and’. In general:
Syntax Rule: Conjunction
If $\phi$ and $\psi$ are formulas, then $[\phi \land \psi]$ is also a formula. (This is called a CONJUNCTION. $\phi$ and $\psi$ are called CONJUNCTS.)

Now, this $\land$ symbol should be interpreted in such a way that $[P \land Q]$ is true whenever both $P$ and $Q$ are true. There are four possibilities: both are true; $P$ is true and $Q$ is false; $P$ is false and $Q$ is true; and both are false. Only in the first case should the conjunction be seen as true. This interpretation of $\land$ can thus be represented as follows:

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

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

Semantic Rule: Conjunction
If $\phi$ and $\psi$ are formulas, then $[\phi \land \psi]^I = 1$ if $[\phi]^I = 1$ and $[\psi]^I = 1$, and 0 otherwise.

The disjunction of $\phi$ and $\psi$ is written $[\phi \lor \psi]$. In such a formula, $\phi$ and $\psi$ are called DISJUNCTS. For example:

$[P \lor Q]$

can be read ‘$P$ or $Q$’. In general:
Syntax Rule: Disjunction
If $\phi$ is a formula and $\psi$ is a formula, then $[\phi \lor \psi]$ is also a formula.

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

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</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>

Semantic Rule: Disjunction
If $\phi$ and $\psi$ are formulas, then $[\phi \lor \psi]^I = 1$ if $[\phi]^I = 1$ or $[\psi]^I = 1$ (or both), and 0 otherwise.

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

(5) Susan volunteers on Monday or Wednesday.

Would you conclude that Susan volunteers on Monday or Wednesday, but not both? If so, then you are getting a so-called EXCLUSIVE interpretation, where the possibility that she volunteers on both days is excluded. An INCLUSIVE interpretation would be one on which the sentence is still true if she volunteers on both days.

The word ‘or’ in English sometimes more naturally has an inclusive interpretation, and sometimes more naturally 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 negated sentence like *I
First-order logic

don't date men or women, it is more likely that an inclusive interpretation is intended, where the speaker dates neither men nor women.

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

Exercise 4. Specify appropriate syntax and semantics rules and an appropriate truth table for the exclusive disjunction connective XOR.

One might imagine that natural language or is ambiguous between inclusive and exclusive disjunction. But there is reason to believe that inclusive reading is what or really denotes, and that the exclusive reading arises via a conversational implicature in certain contexts. One argument for this comes from the fact that negation reliably brings out the inclusive disjunction (e.g. [Horn 1985, Schwarz et al. 2008]. If I say Maria asked Bob not to invite Fred or Sam, it means that Maria asked Bob not to invite Fred and she asked him not to invite Sam. We can get this interpretation by negating the inclusive interpretation (‘it is not the case that at least one of the disjuncts is true’) but not by negating the exclusive one (‘it is not the case that exactly one of the disjuncts is true, i.e. either neither or both are 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. The implicature analysis is also supported by experimental evidence. A classical result, [Paris 1973], has generally found a preference for inclusive interpretations even in unembedded contexts. More recently, [Chevallier et al. 2008] have found that the core meaning of disjunction is inclusive, and that an exclusive in-
interpretation only arises when participants were forced to focus on the disjunction for at least three seconds.

So far, we have discussed the semantics of $\land$, $\lor$, and $\neg$. The truth tables that we have used to define the meaning of the connectives can be strung together to calculate the meaning of complex formulas. For instance, let us consider when the formula $\neg[P \land Q]$ 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>

(The brackets $[ ]$ are crucial here, as they show that we are applying negation to the conjunction of $P$ and $Q$, rather than $P$.) Notice that the final column in the truth table above, for $\neg[P \land Q]$, is the result of ‘flipping’ the truth values in the preceding column, for $P \land Q$. This is what the truth table for negation tells us to do.

With these tools, we can prove equivalences. To prove an equivalence between two formulas using a truth table, construct columns for both formulas, and observe that the two columns have exactly the same pattern of trues and falses; they both true under exactly the same circumstances, and they are both false under the same circumstances. For example, to prove that $P$ is equivalent to $\neg\neg P$, we can use the following truth table, where the columns for the two formulas in question are highlighted:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\neg P$</th>
<th>$\neg\neg P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Since this formula only has one propositional letter, we only need to consider two cases: the case where it’s true (the first row of truth values) and the case where it’s false (the second row). Observe that in the case where \( P \) is true, \( \neg\neg P \) is also true, and in the case where \( P \) is false, \( \neg\neg P \) is also false. If two formulas have the same truth values under any possible interpretation (i.e., in every row of the truth table), they are **equivalent**.

DeMorgan’s laws involve two variables, so there are four cases to consider, as each variable might be either true or false. For instance, to prove that \( \neg[P \land Q] \) is equivalent to \( [\neg P \lor \neg Q] \), let us use the following truth table, where the columns for \( \neg[P \land Q] \) and \( [\neg P \lor \neg Q] \) are highlighted. (The non-highlighted columns are there as intermediate steps that will allow you to compute the highlighted columns, which are the main ones of interest.) As you can see, the pattern of 0s and 1s in the two columns is the same.

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

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

**Exercise 5.** DeMorgan’s Law II: Show that \( \neg[P \lor Q] \) is equivalent to \( [\neg P \land \neg Q] \), using a truth table. Here is a start:
What should we observe about your truth table? In other words, what shows that the two formulas are equivalent?

Entailment can also be proven using truth tables. Recall the definition of entailment: \(\text{A entails B if and only if Whenever A is true, B is also true.}\) Truth tables list out various alternative possible scenarios, and each row of the truth table corresponds to a different imaginable scenario. So we can make the meaning of ‘whenever’ precise using truth tables as follows: \(\text{A entails B if and only if In every row of the truth table where A is true, B is true too.}\) Example:

\[
[P \land Q]
\]

entails

\(P\)

because in every row where \([P \land Q]\) is true, \(P\) is true too. (Well, there is only one row where \([P \land Q]\) is true, namely the top row, but in that row, \(P\) is true, too.)

Exercise 7. Decide whether or not:

$$\neg[P \lor Q]$$

entails

$$\neg[P \land Q]$$

Start by filling in this truth table:

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

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

Before we move on to the remaining connectives, let us note
that truth tables can also help to shed light on **scopal ambiguity**. The following sentence is scopally ambiguous:

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

It can mean either of the following:

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

The two readings can be explained based on the relative scope of negation and conjunction. Assume the following basic translations of English sentences into propositional letters:

\[ P = \text{Geordi consulted Troi}. \quad R = \text{Geordi consulted Worf}. \]

The two readings can be represented as:

(8) a. \(\neg[P \land R]\)
    b. \([\neg P \land \neg R]\)

Note that the second reading is equivalent to \(\neg[P \lor R]\), which could explain why ‘Geordi didn’t consult Troi and Worf’ can mean the same thing as ‘Geordi didn’t consult Troi or Worf’!

**Exercise 8.**

(a) Which of the formulae in (8) captures the reading in (7a)?

(b) Which corresponds to the reading in (7b)?

In general, the outer square brackets with binary connectives are always there according to the official rules of the syntax. We will sometimes drop them when they are not necessary for disambiguation. Sometimes, operator precedence rules are assumed.
For example, in the absence of brackets negation is taken to take scope under (i.e. bind more strongly than) the binary connectives. (The scope of a connective in a formula is the part of the formula that matches the metavariable(s) in its syntactic rule.) This means that a formula like $\neg P \land R$ is the conjunction of $\neg P$ with $R$, and is not equivalent to $\neg [P \land R]$. Likewise, material implication and the biconditional, which we are about to encounter, are sometimes taken to take scope over all other connectives. Brackets can be left in place to either override or reinforce these conventions.

3.2.3 Conditionals and biconditionals

Recall from above the fact that we want our logic to be validate Modus Ponens (‘If $P$ then $Q$; $P$; therefore $Q$’) as an argument form, but not Asserting the Consequent (‘If $P$ then $Q$; $Q$; therefore $P$’). There is a way of defining the semantics of conditional statements (statements of the form ‘if $A$ then $B$’) using truth tables that captures these facts. This method involves the so-called material conditional, a connective written as $\to$ (sometimes also $\supset$).

A formula of the form $[P \to Q]$ is false only when $P$ is true and $Q$ is false, and true otherwise. The truth table for this connective looks like this:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$[P \to Q]$</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>

To get the intuition behind this as a meaning for ‘if . . . then’, consider the following conditional sentence:
If it’s sunny, then it’s warm.

(Terminological note: A conditional sentence is always of the form ‘if A then B’, where A is called the antecedent and B is called the consequent. Here the antecedent is it’s sunny and the consequent is it’s warm.) There are four types of situations, in principle:

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

Let us consider which of these situations would falsify (9). Certainly the first situation does not. And if it’s not sunny, then whether it’s warm is irrelevant, because the claim only pertains to situations where it’s sunny. So the third and the fourth situations would not falsify it. The only kind of situation that could falsify the claim is the second one, where the antecedent is true and the consequent is false.

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 treat the semantics of conditionals as a function of the truth of the antecedent and consequent, material implication as we have
defined it comes closest to doing the job. With material implication, we can validate Modus Ponens and invalidate Denying the Antecedent, for example.

**Exercise 9.** Specify the syntax and semantics rules for the material conditional.

**Exercise 10.** Fun fact: \([P \rightarrow Q]\) is equivalent to \([\neg P \lor Q]\). Show this by filling in the following truth table.

<table>
<thead>
<tr>
<th>(P)</th>
<th>(Q)</th>
<th>([P \rightarrow Q])</th>
<th>(\neg P)</th>
<th>([\neg P \lor Q])</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

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

**Exercise 11.** Let us consider the question of whether **Modus Tol-**

**lens** \((P \rightarrow Q; \neg Q; \text{therefore} \neg P)\) turns out to be a valid argument form. Start by filling in this truth table:

<table>
<thead>
<tr>
<th>(P)</th>
<th>(Q)</th>
<th>([P \rightarrow Q])</th>
<th>(\neg Q)</th>
<th>(\neg P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To determine whether the argument is predicted to be valid, we need to determine whether the conclusion of the argument is true in every case where all of the premises are true. So first, we need to determine in what cases all of the premises are true. There are two premises in the Modus Tollens argument: $P \rightarrow Q$ and $\neg Q$. The first step is to identify the row(s) in which both of these premises are true. The next step is to consider whether the conclusion of the argument ($\neg P$) is true in every such row.

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

**Exercise 12.** Recall that Denying the Antecedent has the form:

Premise 1: $P \rightarrow Q$
Premise 2: $\neg P$
Conclusion: $\neg Q$

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

**Exercise 13.** Consider the sentence in (10).

(10) There is homework due next week unless it is the last week of class.

Assume the following basic translations of English sentences:

\[ P = \text{There is homework due next week.} \]
\[ Q = \text{It is the last week of class.} \]
We currently have no connective corresponding to *unless* in our vocabulary. So let us introduce a new connective, \( \neg \), such that (10) gets the translation in (11).

\[
(11) \quad [P \neg Q]
\]

First, give the semantics of this connective by giving a truth table for it. In other words, fill in the values under \([P \neg Q]\) below:

<table>
<thead>
<tr>
<th>(P)</th>
<th>(Q)</th>
<th>([P \neg Q])</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td><em>(Homework due next week and last week of class)</em></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td><em>(No homework due next week and last week of class)</em></td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td><em>(Homework due next week and not last week of class)</em></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td><em>(No homework due next week and not last week of class)</em></td>
</tr>
</tbody>
</table>

Now, some scholars have argued that ‘*A unless B*’ is equivalent to ‘*A if not B*’. Is that equivalence predicted under the semantics you have given? Illustrate by extending your truth table.

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

\[
(12) \quad \begin{align*}
\text{a. } & \text{If today is Monday, then yesterday was Sunday.} \\
\text{b. } & \text{If yesterday was Sunday, then today is Monday.}
\end{align*}
\]

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

\[
(13) \quad \text{Today is Monday if and only if yesterday was Sunday.}
\]

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

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

This truth table differs from that for \([P \rightarrow Q]\) only in the third row. When \(P\) is false and \(Q\) is true, \([P \rightarrow Q]\) is true but \([P \leftrightarrow Q]\) is false.

**Exercise 14.** Specify the syntax and semantics rules for the biconditional.

### 3.2.4 Equivalence, contradiction and tautology

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

<table>
<thead>
<tr>
<th></th>
<th>(\neg P)</th>
<th>(\neg \neg P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</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 15.** Using truth tables, check whether the following pairs of formulas are equivalent.

(a) \([P ∨ Q]; \neg[\neg P ∧ \neg Q]\)

(b) \([P → Q]; [\neg P ∨ Q]\)

(c) \(\neg[ P ∧ Q]; [\neg P ∨ \neg Q]\)

(d) \([P ∨ \neg Q]; \neg[ P ∧ \neg Q]\)

(e) \([P → Q]; [\neg Q → \neg P]\)

(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.

<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>

Another contradictory pair is \([P → Q]\) and \([P ∧ \neg Q]\).
A tautology (also called valid formula) is a formula that is true under every assignment. The opposite, an expression that is false under every assignment, is called a contradiction; such a formula is also called inconsistent or unsatisfiable. Formulas that are neither valid nor inconsistent are called contingent, and formulas that are either valid or contingent are called satisfiable. You can tell which of these categories a formula falls under by looking at the pattern of 1s and 0s in the column underneath it in a truth table: If they are all true, the formula is satisfiable and valid; if some are true and others are false, it is satisfiable and contingent; otherwise, it is inconsistent. Here is a tautology: \( P \lor \neg P \) (e.g. It is raining or it is not raining):

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>( P )</th>
<th>( \neg P )</th>
<th>( \neg \neg P )</th>
<th>( [ P \leftrightarrow \neg \neg P ] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Exercise 16. Which of the following are tautologies?

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

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

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

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

Support your answer with truth tables.

3.3 Summary: Propositional logic

To summarize what we have covered so far, let us define a simple propositional logic language called \(L_{\text{Prop}}\). All languages of propositional logic are like this language up to the choice of propositional letters. 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 \(L_{\text{Prop}}\)

1. Atomic formulas

   - Propositional letters: \(P, Q, R\)

2. Complex formulas

   - Negation (Unary connective): If \(\phi\) is a formula, then \(\neg \phi\) is a formula.
• **Binary connectives**: If $\phi$ and $\psi$ are formulas, 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 $\psi$’

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

### 3.3.2 Semantics of $L_{\text{Prop}}$

Let $[[\phi]]^I$ stand for the truth value of any expression $\phi$ with respect to an interpretation function $I$ whose domain is the set of propositional letters and whose codomain is the set of truth values $\{0, 1\}$.

1. **Propositional letters**

   - If $\phi$ is any propositional letter, then
     
     $$[[\phi]]^I = I(\phi).$$

2. **Complex formulas**

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

3. **Binary connectives** If $\phi$ and $\psi$ are formulas, then:

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


3.4 From propositional logic to predicate logic

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

But in another respect, propositional logic is much simpler than natural language. While we might represent *Abelard is happy and Eloise is sad* as \([P \land Q]\), and its first conjunct *Abelard is happy* as \(P\), we cannot break it down further. There is nothing in propositional logic that corresponds to *Abelard* or to *happy*. More generally, in propositional logic an atomic formula consists of just one propositional letter, but natural sentences that atomic formulas represent can consist of multiple words.

To remedy this, we will now “split the atom”. This corresponds to the point where we go beyond the resources of propositional logic and move into what will turn out to be PREDICATE LOGIC, or more specifically FIRST-ORDER LOGIC. The propositional letters and connectives of propositional logic all carry over to first-order logic. But in first-order logic, atomic formulas may be built up of several BASIC EXPRESSIONS—symbols of the language with no internal structure: names like \(a\) (for Abelard) and \(e\) (for Eloise), predicates like \(\text{Happy}\) and \(\text{Loves}\), and functions like \(\text{motherOf}\). Con-

---

\(^2\)Peter Abelard was a philosopher and theologian in 12th century France, arguably the greatest logician of the Middle Ages and an important thinker on reason and religion. His affair with Eloise, already a reputed scholar when she became his student, led to their secret marriage and, tragically, to his castration. At that point, Abelard became a monk, and Eloise a nun (and eventually a prioress). Their correspondence is among the most moving and personal documents of the 12th century.
strained by the syntax rules that we will define for the language, these basic expressions may be put together in various ways to form atomic formulas. First-order logic is also characterized by the presence of VARIABLES and QUANTIFIERS, which we will delay until Section 3.6.

### 3.4.1 Individual constants

In first-order logic, individual objects are named by INDIVIDUAL CONSTANTS or NAMES. Individual constants are an example of TERMS. A term is an expression that refer to individuals (rather than denoting a set or a relation or a truth value). Aside from constants, terms also include VARIABLES, as we will see later. 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,

$$s_j$$

is a valid individual constant, but

$$S$$

is not, nor is

$$s \text{ and } j.$$  

Recall that in propositional logic, expressions are interpreted relative to an interpretation function $I$, which maps propositional letters to truth values. In first-order logic, this interpretation function is given additional tasks. One of them is to map individual constants to individuals (which may be thought of as real-world objects). The set of individuals that are available for this purpose is called the DOMAIN, which we will write as $D$. The pair $(D, I)$ is called a MODEL, and we will write it as $M$.

To illustrate, we will define a language $L_0$ and interpret it in models whose domain is $D_0$. This language will allow us to talk about the four members of the Swedish pop band ABBA, whose
names are *Agneta*, *Björn*, *Benny*, and *Anni-Frid*. Our language $L_0$ should contain expressions that refer to these individuals. Let us assume that expressions in $L_0$ may contain the following individual constants: $ag$, $bj$, $be$, $an$. Let us also assume that our interpretation functions map these constants to the appropriate band members. For example, we might define a model $M_0 = \langle D_0, I_0 \rangle$ and define $I_0$ as follows:

\[
\begin{align*}
& a. \quad I_0(\text{ag}) = \text{Agneta} \\
& b. \quad I_0(\text{bj}) = \text{Björn} \\
& c. \quad I_0(\text{be}) = \text{Benny} \\
& d. \quad I_0(\text{an}) = \text{Anni-Frid}
\end{align*}
\]

Unlike names in English such as *Björn*, individual constants are not ambiguous; they pick out exactly one object. However, not every individual object in a given model needs to have a corresponding individual constant in a given language; we might have any number of objects in the model’s domain without any individual constant in the language. 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.

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

\[
\begin{align*}
& a. \quad [\text{ag}]^M_0 = \text{Agneta} \\
& b. \quad [\text{bj}]^M_0 = \text{Björn} \\
& c. \quad [\text{be}]^M_0 = \text{Benny} \\
& d. \quad [\text{an}]^M_0 = \text{Anni-Frid}
\end{align*}
\]

The following rule ensures that the $[\cdot]$ function tracks the $I$ function on individual constants.
Semantic Rule: Non-logical constants
If $\alpha$ is a non-logical constant and $M = \langle D, I \rangle$, then:

$$\llbracket \alpha \rrbracket^M = I(\alpha)$$

Like $\phi$ and $\psi$ in Section 3.2, $\alpha$ is a metavariable. It is not part of the language $L_0$ we are defining but only part of the metalanguage we are using to talk about that language. We will see other examples of metavariables later on.

This rule is formulated in reference not to individual constants but to non-logical constants. This category includes not only individual constants but also some additional types of symbols that will be introduced below: predicate and function symbols. Non-logical constants are called this way because their denotation is determined by the interpretation function $I$ and can therefore vary from model to model. The interpretation function decides, for example, who has which name, who loves whom, who is happy, and generally what all of the non-logical constants denote. The term 'constant symbol' stands in opposition to 'variable'. We will introduce variables later; for now, you may just think of constants as the basic expressions of the language. The logical constants, in contrast, include the connectives $\land$, $\lor$, $\neg$, $\rightarrow$, and $\leftrightarrow$; these have the same meaning in every model.

We refer to the actual members of ABBA in our meta-language using their first names, and we write them capitalized and in ordinary type face. To echo Dowty et al. (1981):

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

If it had been possible to persuade Agneta to come and occupy the right-hand side of the equation above for a moment, that would certainly have been preferable, but this the closest we can come using the printed page.

In Chapter 5, we will define a system that relates English expressions to logical expressions like this, and thereby indirectly associate English expressions with denotations in the model. The mapping between expressions of the natural language (English) and their denotations (expressed in our meta-language) will thus be mediated by our logical representation language. So our ultimate theory will consist of two steps:

- $\text{Björn} \rightsquigarrow \text{bj}$ \hspace{1cm} (English to logic)
- $\llbracket \text{bj} \rrbracket^M_0 = \text{Björn}$ \hspace{1cm} (logic to meaning)

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

- $\text{female} \rightsquigarrow \text{Female}$ \hspace{1cm} (English to logic)
- $\llbracket \text{Female} \rrbracket^M_0 = \{\text{Agneta, Anni-Frid}\}$ \hspace{1cm} (logic to meaning)

The style of doing semantics we are adopting here is called INDIRECT INTERPRETATION. There is another style, known as DIRECT INTERPRETATION, where the mapping is carried out in one fell swoop:
• $\text{[Björn]} = \text{Björn}$ (English to meaning)

• $\text{[female]}^{M_0} = \{\text{Agneta}, \text{Anni-Frid}\}$ (English to meaning)

Both styles are common in semantic theory. The direct interpretation style is used in the classic Heim & Kratzer (1998) textbook. However, indirect interpretation carries a number of practical advantages, which we will point out along the way.

3.4.2 Predication

3.4.2.1 Syntax of predication

Along with individual constants, first-order logic has predicates. Intuitively, these are expressions that pick out sets of objects. A predicate can be thought of as consisting of a symbol (a string of letters, such as like Female, Swedish or Sings) that denotes a set (such as the set of female or Swedish or singing individuals). Sometimes, the term predicate symbol is used to refer to the symbol, and the term predicate is then used just for the set that it denotes. While there is no standard convention, in this textbook, we will always capitalize the first letter of each predicate symbol of first-order logic.

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

(16) \text{Swedish}(ag)

will be interpreted as saying that Agneta is Swedish. We say that the predicate Swedish holds of or applies to Agneta. The complex consisting of the predicate with its argument is called a formula, because it expresses a proposition, something that can be true or false. We will encounter other types of formulas later on.

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

(17) \text{Loves}(ag, bj)

We say that the predicate \text{Loves} holds of, applies to, or relates Agneta and Björn.

The number of arguments that a predicate takes is its \textit{arity}, also called \textit{valence} or \textit{adicity}. Unary predicates take one argument, and therefore have an arity of 1. Binary predicates have an arity of 2. A \textit{ternary predicate} has an arity of 3. As an example, we might defined a ternary predicate \text{Between}, as holding of three objects \(x, y\) and \(z\), if \(x\) is between \(y\) and \(z\). There is no upper limit to the arity of predicates. Sometimes it is useful to regard propositional letters as \textit{zero-place} or \textit{nullary} predicates. It is also common to speak of \textit{one-place} or \textit{monadic}, \textit{two-place} or \textit{dyadic}, and generally of \textit{n-ary}, \textit{n-place} or \textit{n-adic predicates}.

In first-order logic, the arity of a predicate is fixed. The arity of corresponding natural language expressions is much more free; for example, English allows the adjective \textit{proud} to take a prepositional phrase complement but does not require it to.

(18) a. Agneta is proud of Benny.
    b. Agneta is proud.

In first-order logic, a predicate like \textit{Proud} may only have one arity; it cannot be both unary and binary. To represent the difference between the transitive and intransitive version of \textit{proud} in English, one option would be to define two predicates, say, a unary predicate \textit{Proud} and a binary predicate \textit{ProudOf}. To capture how close in meaning these two predicates otherwise are, a theory could stipulate facts about how they relate to each other via constraints that are stipulated separately. (See meaning postulates below.)

(19) a. \text{ProudOf}(ag, be)
b. Proud\((ag)\)

Another option would be to define just the unary predicate Proud and a binary predicate Of, and to use conjunction:

\[
\begin{align*}
\text{(20) a. } & \text{Proud\((ag)\)} \land \text{Of\((ag, be)\)} \\
\text{b. } & \text{Proud\((ag)\)}
\end{align*}
\]

We will come back to this issue in Section \[3.6\].

In general, predicates combine with the appropriate number of arguments to form \textsc{atomic formulas}. For example,

\[
\text{Singer\((ag)\)}
\]

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

\[
\text{Loves\((ag, bj)\)}
\]

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

\[\text{Syntax Rule: Predication}\]

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.

(A summary of definitions like this will be compiled at the end of this section.)
Exercise 17. Give two examples of atomic formulas generated by the syntax rule of Predication, choosing from the following individual constants and predicates:

- \( ag \) and \( bj \) are individual constants (a.k.a. ‘names’);
- \( \text{Singer} \) and \( \text{Swedish} \) are one-place predicates;
- \( \text{Knows} \) and \( \text{Loves} \) are two-place predicates.

3.4.2.2 Semantics of predication

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

\[
\begin{align*}
(21) & \quad \text{a. } \text{Loves}(ag, bj) \\
& \quad \text{b. } \text{Loves}(bj, ag)
\end{align*}
\]

\[
\begin{align*}
(22) & \quad \text{a. } \text{Loves}(be, an) \\
& \quad \text{b. } \text{Loves}(an, be)
\end{align*}
\]

Now, as it happens, like all good things, both of the marriages eventually came to an end, and these four statements, concomitantly, ceased to be true (we assume). So far, we have only had one model, \( M_{ABBA} \). To distinguish between the way it was in the past, and how things later turned out, we will now extend it in two different ways: \( M_{\text{THEN}} \) corresponds to how it was back in the day, and \( M_{\text{NOW}} \) to how it is now. These two models share the same domain, \( D_0 \), but their interpretation functions will differ. We will define

\[
M_{\text{THEN}} = \langle D_0, I_{\text{THEN}} \rangle \quad \text{and} \quad M_{\text{NOW}} = \langle D_0, I_{\text{NOW}} \rangle.
\]
Relative to these two different models, the binary predicate \textit{Loves} has two different semantic values. Accordingly, we make the following assumptions about \( I_{\text{THEN}} \) and \( I_{\text{NOW}} \):

\[
(23) \quad I_{\text{THEN}}(\text{Loves}) = \\
\{ (\text{Agneta}, \text{Björn}), (\text{Björn}, \text{Agneta}), \\
(\text{Anni-Frid}, \text{Benny}), (\text{Benny}, \text{Anni-Frid}) \}
\]

\[
(24) \quad I_{\text{NOW}}(\text{Loves}) = \{ \}
\]

That is, back in the day, Agneta and Björn loved each other, and so did Benny and Anni-Frid, but now, nobody loves each other.

Just as we did for individual constants, we need to make sure that \([\cdot] \) function tracks the \( I \) function on predicates. We do this by declaring predicates to be non-logical constants. This means that the same semantic rule we introduced for individual constants can be reused. It is repeated here:

**Semantic Rule: Non-logical constants**

If \( \alpha \) is a non-logical constant and \( M = (D, I) \), then:

\[
\llbracket \alpha \rrbracket^M = I(\alpha)
\]

Just as we did for propositional letters, we will assume that the denotation of a formula like \( \text{Singer}(ag) \) is a truth value:

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

The denotations of unary predicates may of course differ across models. For example, in some models, Anni-Frid is a singer, and in other models, she stays off the microphone and sticks to the synthesizer. Whether or not she is in the extension of the unary predicate \( \text{Singer} \), and hence, whether the formula \( \text{Singer}(an) \) is true, depends on whether the denotation of an (namely Anni-Frid) is in the set denoted by \( \text{Singer} \). In general, for any given unary
predicate $\pi$ and any given term $\alpha$, we would like the semantics of our language to ensure the following:

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

This can be read, “the semantic value of $pi$ applied to $alpha$ (with respect to model $M$) is 1, if the semantic value of $alpha$ (with respect to $M$) is an element of the semantic value of $pi$ (with respect to $M$), and 0 otherwise.” To put it somewhat more elegantly: “Relative to any given model, the predication of $\pi$ upon $\alpha$ is true in that model if and only if the denotation of $\alpha$ in that model is a member of the set denoted by $\pi$ in that model.”

Our semantics should also ensure that the formula $\text{Loves}(ag, bj)$ is true relative to $M_{\text{THEN}}$ and false relative to $M_{\text{NOW}}$.

(25) a. $\left[ \text{Loves}(ag, bj) \right]^{M_{\text{THEN}}} = 1$
   because $(\text{Agneta, Björn}) \in \left[ \text{Loves} \right]^{M_{\text{THEN}}}$

b. $\left[ \text{Loves}(ag, bj) \right]^{M_{\text{NOW}}} = 0$
   because $(\text{Agneta, Björn}) \notin \left[ \text{Loves} \right]^{M_{\text{NOW}}}$

In general, for any binary predicate $\pi$, and any given terms $\alpha$ and $\beta$, our semantics should ensure:

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

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

**Semantic Rule: Predication**

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

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

To make sure this works as expected in the unary case, we adopt the convention that $\left(\left[ \alpha_n \right]^M\right) = \left[ \alpha_n \right]^M$. 
Exercise 18. Suppose we have a particular model $M = \langle D, I \rangle$. Let $D = \{\text{Agneta, Björn, Benny, Anni-Frid}\}$. Suppose that in $M$, everybody loves themselves and nobody loves anybody else, and the binary predicate Loves denotes the love relation. What is then the value of $I(\text{Loves})$? Specify the relation as a set of ordered pairs.

Exercise 19. Assume a model

$$M = \langle D, I \rangle$$

where $D$ contains Abelard and Eloise:

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

and $I$ is defined as follows:

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

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

(a) Teacher(a,e)
(b) Teacher(e,a)
(c) Teacher(e,a)
(d) Female(a)
(e) Loves(a,a)
3.4.3 Functions

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

\[ \text{spouseOf(be)} \]

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

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

\[ \text{Loves(spouseOf(be), be)} \]

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

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

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

\[ \text{tallerOf(ag,an)} \]
would return Anni-Frid, if she is the taller of the two. In general:

**Syntax Rule: Complex terms**
Given any function $\gamma$ with arity $n$, then:

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

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

Function symbols denote functions (of the kind discussed in the previous chapter, namely, relations of a special kind); this denotation is specified by the interpretation function $I$. Functions are considered non-logical constants; therefore, their denotation is derived from $I$ according to the same rule as individual constants and predicates. However, functions combine with their arguments in a slightly different manner from the way predicates do. The denotation of a function symbol applied to a term is the result of applying the function denoted by the function symbol to the denotation of the term. For example, suppose that in model $M$, spouseOf denotes a function that gives Anni-Frid when given the individual Benny as an argument. Then $\text{spouseOf}(\text{be})$ denotes Anni-Frid, i.e., $\left[\text{spouseOf}(\text{be})\right]^M = \text{Anni-Frid}$. In general:

**Semantic Rule: Complex terms**
If $\gamma$ is a unary function symbol, and $\alpha$ is a term, then:

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

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 exam-
ple, the following expression:

\[
\text{tallerOf}(\text{ag, an})
\]

would denote the result of applying the function denoted by \text{tallerOf}
to the ordered pair \langle\text{Agneta, Anni-Frid}\rangle. In general:

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

\[
\langle{}^{\downarrow} \gamma(\alpha, \beta) \rangle^M = \langle{}^{\downarrow} \gamma \rangle^M(\langle{}^{\downarrow} \alpha \rangle^M, \langle{}^{\downarrow} \beta \rangle^M)
\]

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

**Semantic Rule: Atomic formulas**
If \(\gamma\) is a function of arity \(n\), and \(\alpha_1, ..., \alpha_n\) is a sequence of \(n\) terms,
then:

\[
\langle{}^{\downarrow} \gamma(\alpha_1, ..., \alpha_n) \rangle^M = \langle{}^{\downarrow} \gamma \rangle^M(\langle{}^{\downarrow} \alpha_1 \rangle^M, ..., \langle{}^{\downarrow} \alpha_n \rangle^M)
\]

### 3.4.4 Equality

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

\[
\text{ag = be}
\]

\[
\text{spouseOf(ag) = bj}
\]

\[
\text{an = tallerOf(an, ag)}
\]
The following rule dictates that any two terms, simple or complex, can be joined in this way to form an atomic formula:

**Syntactic Rule: Equality**
If \( \alpha \) and \( \beta \) are terms, then \( \alpha = \beta \) is an atomic formula.

The corresponding semantic rule is as you might expect:

**Semantic Rule: 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.

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

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>individual constants</td>
<td>ag</td>
</tr>
<tr>
<td>unary predicates</td>
<td>Singer</td>
</tr>
<tr>
<td>binary predicates</td>
<td>Loves</td>
</tr>
<tr>
<td>function symbols</td>
<td>spouseOf</td>
</tr>
</tbody>
</table>

The interpretation function of a model determines the semantic values of these constant symbols. Atomic formulas can be produced either through predication or equality.
Exercise 20. Let us consider a model $M = (D, I)$ with domain $D$ consisting only of two individuals: Abelard and Eloise. Let us assume that among our basic expressions we have names for both Abelard and Eloise (say $a$ and $e$ respectively), as well as the unary predicates Scholar, Male, and Female, binary predicates Loves and Teacher, and the function terms loverOf and self. Fill in the missing values in the interpretation function, according to what you think they should be based on the constant symbol:

$I = \begin{align*}
    a & \rightarrow \text{Abelard} \\
    e & \rightarrow \text{Eloise} \\
    \text{Female} & \rightarrow \{\text{Eloise}\} \\
    \text{Male} & \\
    \text{Scholar} & \rightarrow \{\text{Abelard, Eloise}\} \\
    \text{Loves} & \rightarrow \{(\text{Abelard, Eloise}), (\text{Eloise, Abelard}), (\text{Eloise, Eloise})\} \\
    \text{Teacher} & \rightarrow \{(\text{Abelard, Eloise})\} \\
    \text{YoungerThan} & \\
    \text{loverOf} & \rightarrow \{(\text{Abelard, Eloise}), (\text{Eloise, Abelard})\} \\
    \text{self} &
\end{align*}$

Exercise 21. Fill in the following table.

<table>
<thead>
<tr>
<th>Term or formula?</th>
<th>$[.]^M$</th>
<th>Semantic Rule(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>loverOf($a$)</td>
<td>term</td>
<td>Eloise</td>
</tr>
<tr>
<td>Female($a$)</td>
<td>formula</td>
<td>0</td>
</tr>
<tr>
<td>Male($a, e$)</td>
<td>not well-formed!</td>
<td>N/A</td>
</tr>
<tr>
<td>Teacher($a, e$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teacher(loverOf($e$), e)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loves($a, self(a)$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>loverOf(loverOf($e$))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scholar(self(self($a$)))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scholar($a, self(a)$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teacher(self(self($a$)))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In the first labelled column, state whether the expression is a term, a formula, or not well-formed. In the second, give the semantic value relative to the model you designed in exercise 20. In the third, indicate the semantic rule(s) you used to derive the semantic value in the second column.

3.5 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.

3.5.1 Syntax of \( L_0 \)

1. Basic Expressions
   - Individual constants: \( ag, be, bj, an \)
   - Function symbols
     - Unary: \( \text{spouseOf} \)
     - Binary: \( \text{tallerOf} \)
   - Predicates
     - Unary: \( \text{Swedish}, \text{Singer} \)
     - Binary: \( \text{Loves} \)

2. Terms
   - Names: Every individual constant is a term.
• **Complex terms**: If $\pi$ is a function symbol of arity $n$, and $\alpha_1, \ldots, \alpha_n$ are terms, then $\pi(\alpha_1, \ldots, \alpha_n)$ is a term\(^3\).

3. **Atomic formulas**

• **Predication**: If $\pi$ is a function of arity $n$ and $\alpha_1, \ldots, \alpha_n$ are terms, then $\pi(\alpha_1, \ldots, \alpha_n)$ is an atomic formula\(^4\).

• **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 $\psi$’

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.

\(^3\)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.

\(^4\)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.
3.5.2 Semantics of $L_0$

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

1. **Basic expressions**
   - If $\alpha$ is a non-logical constant (individual constant, predicate, or function), then $\sem{\alpha}^M = I(\alpha)$.

2. **Complex terms**
   - If $\pi$ is a function of arity $n$, and $\alpha_1,...,\alpha_n$ is a sequence of $n$ terms, then $\sem{\pi(\alpha_1,...,\alpha_n)}^M = \sem{\pi}^M(\sem{\alpha_1}^M,...,\sem{\alpha_n}^M)$.

3. **Atomic formulas**
   - **Predication**: If $\pi$ is a predicate of arity $n$, and $\alpha_1,...,\alpha_n$ is a sequence of $n$ terms, then $\sem{\pi(\alpha_1,...,\alpha_n)}^M = 1$ if $\langle \sem{\alpha_1}^M,...,\sem{\alpha_n}^M \rangle \in \sem{\pi}^M$, and $0$ otherwise.
   - **Equality**: If $\alpha$ and $\beta$ are terms, then $\sem{\alpha = \beta}^M = 1$ if $\sem{\alpha}^M = \sem{\beta}^M$, and $0$ otherwise.

4. **Negation**
   - $\sem{\neg \phi}^M = 1$ if $\sem{\phi}^M = 0$, and $0$ otherwise.

5. **Binary connectives**

---

$^5$Special cases:
- When $\pi$ is a function of arity 1, then $\sem{\pi(\alpha)}^M = \sem{\pi}^M(\sem{\alpha}^M)$.
- When $\pi$ is a function of arity 2, then $\sem{\pi(\alpha, \beta)}^M = \sem{\pi}^M(\sem{\alpha}^M, \sem{\beta}^M)$.

$^6$Special cases:
- If $\pi$ is a unary predicate and $\alpha$ is a term, then $\sem{\pi(\alpha)}^M = 1$ iff $\sem{\alpha}^M \in \sem{\pi}^M$.
- If $\pi$ is a binary predicate and $\alpha$ and $\beta$ are terms, then $\sem{\pi(\alpha, \beta)}^M = 1$ iff $\langle \sem{\alpha}^M, \sem{\beta}^M \rangle \in \sem{\pi}^M$. 
• \([\phi \land \psi]_M = 1 \) if \([\phi]_M = 1\) and \([\psi]_M = 1\), and 0 otherwise.

• \([\phi \lor \psi]_M = 1 \) if \([\phi]_M = 1\) or \([\psi]_M = 1\), and 0 otherwise.

• \([\phi \rightarrow \psi]_M = 1 \) if \([\phi]_M = 0\) or \([\psi]_M = 1\), and 0 otherwise.

• \([\phi \leftrightarrow \psi]_M = 1 \) if \([\phi]_M = [\psi]_M\), and 0 otherwise.

Notice that we have specified the semantics for the connectives in words rather than using truth tables. Of these, the semantic clause for material implication (\(\rightarrow\)) is worth special attention. We said above that \(\phi \rightarrow \psi\) is false only when \(\phi\) is true and \(\psi\) is false. Does that mean that it is true whenever \(\phi\) is false or \(\psi\) is true? It should, if we have specified the semantics correctly. Readers are encouraged to verify this for themselves.

### 3.6 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 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:

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

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

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

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 UNIVERSAL QUANTIFIER, written \( \forall \). If we had used the universal quantifier instead of the existential quantifier in the formula above, we would have expressed the claim that everything satisfies the condition. Thus 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 Every athlete got an A. It turns out that we can represent this as follows:

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

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:

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

This says, “For all \( x \), \( x \) is an athlete and \( x \) received an A” – in other words, “Everything is an athlete and everything received an A.”

Let us take some time to reflect on why the universally quantified formula with the material implication above expresses the every claim, that every athlete got an A. We will formalize the semantics of universally quantified statements shortly, but intuitively, here is how it works. What this formula expresses is that each element of the domain satisfies the condition:

\[ [\text{Athlete}(x) \rightarrow \text{ReceivedGrade}(x, a)] \]
The semantics for $\forall$ asks us to go through each individual in the domain, and consider what happens when $x$ is interpreted as that individual. There are two types of cases that are important to consider: the value of $x$ is an athlete, or the value of $x$ is not an athlete. Consider a value for $x$ who is not an athlete. For this value of $x$, the condition

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

is not met, so the antecedent is false. By the definition of material implication, this means that the conditional as a whole is true. So any value for $x$ that is not an athlete vacuously satisfies $[\text{Athlete}(x) \rightarrow \text{ReceivedGrade}(x, a)]$. The only kind of value for $x$ that could fail to satisfy this condition would be an athlete that did not get an A. Then the antecedent would be true, and the consequent would be false, so the conditional statement as a whole would be false. If there are no athletes that did not get an A, then the formula is true. And this is exactly what Every athlete got an A says.

Now consider the following formula:

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

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 22.** 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) $$\forall x. \text{Kind}(x)$$

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

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

(d) $$\exists x. [\text{Kind}(x) \lor \text{Happy}(x)]$$
(e) $\forall x. [\text{Kind}(x) \rightarrow \text{Happy}(x)]$

(f) $\forall x. \neg \text{Kind}(x)$

(g) $\exists x. \neg \text{Kind}(x)$

(h) $\neg \exists x. \text{Kind}(x)$

(i) $\forall x. \exists y. \text{Loves}(y, x)$

**Exercise 23.** 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 kind and happy.

(b) Everybody is kind and happy.

(c) Everybody who is kind is happy.

(d) Nobody is kind.

(e) Somebody is not kind.

(f) Somebody is kind or happy.

(g) Everybody loves somebody.

(h) Somebody loves everybody.

**Exercise 24.** Which of the following statements in first-order logic better represents the meaning of *Every cellist smokes*?

(a) $\forall x. [\text{Cellist}(x) \rightarrow \text{Smokes}(x)]$
Exercise 25. Express the following sentences in $L_1$:

(a) There is a red car.
(b) All cars are red or green.
(c) No car is blue.
(d) Alan dislikes all cars.

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

Exercise 26. Express the following sentences in $L_1$. In some cases, there may be quantifier scope ambiguity; in that case, give a representation in $L_1$ corresponding to both interpretations.

(a) Every even number is divisible by two.
(b) Everything has a reason.
(c) Something is the reason for everything.
(d) Every human being has at least two mothers.
(e) All fathers are older than their children.
(f) If a man is a philosopher then he is mortal.
(g) Some statues are not of marble.
(h) All statues are not of marble.
(i) He who sins sleeps badly.

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

**Note:** This exercise is extremely challenging!

Now let us start defining the syntax of this language formally. We will allow an infinite number of variables $v_0, v_1, v_2, \ldots$ 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 $\exists$.

**Syntax Rule for $L_1$: Quantification**

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

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

is a formula, and so is

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

For example, $\forall allx. Happy(x)$ 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. $\forall x[Happy(x) \rightarrow Kind(x)]$. Note: In a formula of the form $\forall u. \phi$ or $\exists u. \phi$, $\phi$ is called the **scope** of the quantifier.

Now for the semantics. We continue to treat models as pairs consisting of a domain and an interpretation function, so a given
model $M$ will be defined as $\langle D, I \rangle$ where $D$ is the set of individuals in the domain of the model, and $I$ is a function giving a value to every non-logical constant in the language. Informally,

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

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:

$$
g_1 = \begin{bmatrix}
x & \rightarrow & \text{Maggie} \\
y & \rightarrow & \text{Bart} \\
z & \rightarrow & \text{Bart} \\
\vdots
\end{bmatrix} \quad g_2 = \begin{bmatrix}
x & \rightarrow & \text{Bart} \\
y & \rightarrow & \text{Homer} \\
z & \rightarrow & \text{Bart} \\
\vdots
\end{bmatrix}
$$

The domain of an assignment function is the set of variables ($v_n$ 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

$$
[[\varphi]]^M
$$
we will now write:

\[ [\phi]^{M,g} \]

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

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

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

(26) \[ [x]^{M,g} = g(x) \]

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

**Exercise 27.** In this exercise, use the assignment functions \( g_1 \) and \( g_2 \) that we defined above.

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

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

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

(d) What is \( [y]^{M,g_2} \) (for any model \( M \))?  

From now on, our semantic denotation brackets will have two superscripts: one for the model, and one for the assignment function. In some cases, the choice of assignment function will not make any difference for the semantic value of the expression. For example, take any model \( M \) in which the constant Happy is defined. \( [\text{Happy}]^{M,g_1} \) will be the same as \( [\text{Happy}]^{M,g_2} \) for any model \( M \) and any two assignments \( g_1 \) and \( g_2 \), because \( \text{Happy} \) is a constant. Since it is a non-logical constant, its semantic value depends on the model, but that is the only thing that it depends on.
In particular, it does not depend on any assignment function. But the value of the formula

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

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

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,

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

\[ g_1[y \mapsto \text{Homer}] \]

would be as follows:

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

We changed it so that \( y \) maps to Homer and kept everything else the same.
Exercise 28.

(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 give the following official semantics for $\exists x. \text{Happy}(x)$:

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

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. In general:

**Semantic Rule: Existential quantification**

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

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:
Semantic Rule: Universal quantification

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

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

3.6.1 Syntax of L₁

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

1. Basic Expressions

   • **Individual constants:** ag, bj, be, an, ...
   
   • **Individual variables:** \( v_n \) for every natural number \( n \)
   
   • **Function symbols**
     
     – Unary: spouseOf, ...
     
     – Binary: tallerOf, ...

   • **Predicate symbols**
     
     – Unary: Happy, ...
     
     – Binary: Loves, ...

2. Terms

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

\[ ^7 \]

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.
3. Atomic formulas

- **Predication**
  If \( \pi \) is a predicate of arity \( n \) and \( \alpha_1, \ldots, \alpha_n \) is a sequence of terms, then \( \pi(\alpha_1, \ldots, \alpha_n) \) is an atomic formula.\footnote{Special cases:}

- **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.

\footnote{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.
• The free variables in \( \phi \) are also free in \( \neg \phi \), and the free variables in \( \phi \) and \( \psi \) are free in \( [\phi \land \psi] \), \( [\phi \lor \psi] \), \( [\phi \rightarrow \psi] \), 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**. A 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:

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

because it is clear that the scope of the existential quantifier does not extend any farther to the right than it does. Furthermore, when reading a formula, you may assume that the scope of a binder (e.g. \( \forall x \) or \( \exists x \)) extends as far to the right as possible. So, for example, \( \forall x. [P(x) \land Q(x)] \) 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 can drop the dot between two quantificational binders in a row. Thus instead of:

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

we can write:

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

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

### 3.6.2 Semantics of \( L_1 \)

Now for the semantics of \( L_1 \). The semantic value of an expression is determined relative to two parameters:

1. a model \( M = \langle D, I \rangle \) where \( D \) is the set of individuals and \( I \) is a function mapping each non-logical constant of the language to an element, subset, or relation over elements in \( D \), depending on the nature of the constant.

2. an assignment function \( g \) mapping each individual variable in \( L_1 \) to some element in \( D \).

For any given model \( M \) and assignment function \( g \), the denotation of a given expression \( \alpha \) relative to \( M \) and \( g \), written \( [\alpha]^{M,g} \), is defined as follows:

1. **Basic Expressions**
   - If \( \alpha \) is a non-logical 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:

$$[[\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 $\alpha_1, ..., \alpha_n$ is a sequence of terms, then: $[[\pi(\alpha_1, ..., \alpha_n)]^M = 1$ if $[[\alpha_1]^M, ..., [[\alpha_n]^M] \in [[\pi]^M$, and 0 otherwise.

- Equality
  If $\alpha$ and $\beta$ are terms, then

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

and 0 otherwise.

4. Negation

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

5. Binary Connectives

- $[[\phi \land \psi]^M,g] = 1$ if $[[\phi]^M,g = 1$ and $[[\psi]^M,g = 1$, and 0 otherwise.

- $[[\phi \lor \psi]^M,g] = 1$ if $[[\phi]^M,g = 1$ or $[[\psi]^M,g = 1$, and 0 otherwise.

---

$^9$Special cases:
- When $\pi$ is a function of arity 1, then:

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

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

$$[[\pi(\alpha, \beta)]^M,g] = [[\pi]^M,g(\{[[\alpha]^M,g, [[\beta]^M,g\}).$$
• $[[\phi \rightarrow \psi]]^{M,g} = 1$ if $[[\phi]]^{M,g} = 0$ or $[[\psi]]^{M,g} = 1$, and 0 otherwise.

• $[[\phi \leftrightarrow \psi]]^{M,g} = 1$ if $[[\phi]]^{M,g} = [[\psi]]^{M,g}$, and 0 otherwise.

6. Quantification

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

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

and 0 otherwise.

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

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

and 0 otherwise.

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 29
Consider the following formulas.

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

**Questions:**

(i) Which of the above are well-formed formulas of \(L_1\)?

(ii) Of the ones that are well formed in \(L_1\), which of the above formulas have free variables in them? (In other words, which of them are open formulas?)

Recommended: Express your answer in the form of a table, with one column for each question.
**Exercise 30.** Consider the following model $M_f = (D, I_f)$, where everybody is happy:

$$I_f(\text{Happy}) = \{\text{Bart, Homer, Maggie}\}$$

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

(a) What is $[x]^{M_f, g_{\text{Bart}}}$? Apply the L₁ semantic interpretation rule for variables.

(b) What is $[\text{Happy}]^{M_f, g_{\text{Bart}}}$? Apply the relevant L₁ 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) Using the rule you identified in your answer to the previous question, explain carefully why $[\text{Happy}(x)]^{M_f, g_{\text{Bart}}} = 1$.

**Exercise 31.** Consider the following four assignment functions.

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

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

For each of the following expressions, give the semantic value of the expression relative to the model $M$ defined in Exercise 20 and each of the four assignment functions, using the syntax and semantics of L₁. In other words, say for each expression $\alpha$ what $[\alpha]^{M,g}$ is, for each given assignment function $g$. 
Give your answer in the form of a table, with columns labelled $g_{ae}$, $g_{ea}$, $g_{ee}$, and $g_{aa}$.

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) $x$</td>
<td>(b) $y$</td>
<td>(c) $a$</td>
<td>(d) $\text{loverOf}(x)$</td>
</tr>
<tr>
<td>(e) $\text{Female}(x)$</td>
<td>(f) $\text{Female}(x) \land \text{Scholar}(x)$</td>
<td>(g) $\text{Female}(x) \rightarrow \text{Scholar}(x)$</td>
<td>(h) $\text{Teacher}(x, y)$</td>
</tr>
<tr>
<td>(i) $\exists y. \text{Teacher}(x, y)$</td>
<td>(j) $\exists x \exists y. \text{Teacher}(x, y)$</td>
<td>(k) $\text{Teacher}(a, y)$</td>
<td>(l) $\exists y. \text{Teacher}(a, y)$</td>
</tr>
<tr>
<td>(m) $\text{Loves}(x, \text{loverOf}(x))$</td>
<td>(n) $\forall x. \text{Loves}(x, \text{loverOf}(x))$</td>
<td>(o) $\text{Loves}(x, y)$</td>
<td>(p) $\forall x. \text{Loves}(x, y)$</td>
</tr>
<tr>
<td>(q) $\exists y \forall x. \text{Loves}(x, y)$</td>
<td>(r) $\text{Teacher}(x, y) \rightarrow \text{Male}(x)$</td>
<td>(s) $\forall y. \text{Teacher}(x, y) \rightarrow \text{Male}(x)$</td>
<td>(t) $\forall x \forall y. \text{Teacher}(x, y) \rightarrow \text{Male}(x)$</td>
</tr>
</tbody>
</table>
Exercise 32. Let $g$ be defined such that $x \mapsto \text{Marge}$, $y \mapsto \text{Bart}$, and $z \mapsto \text{Homer}$, and suppose that in $M_2$, everybody loves themselves and nobody loves anybody else, and the binary predicate $\text{Loves}$ denotes this love relation. Assume that $m$ denotes Marge.

(a) Calculate:

(i) $\lbrack x \rbrack _{M_2, g}$

(ii) $\lbrack m \rbrack _{M_2, g}$

(iii) $\lbrack \text{Loves} \rbrack _{M_2, g}$

(iv) $\lbrack \text{Loves}(x, m) \rbrack _{M_2, g}$

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

(c) For each of those value assignments $g_i$ in the set $\{g_1, \ldots, g_n\}$, calculate $\lbrack \text{Loves}(x, m) \rbrack _{M_2, g_i}$.

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

Exercise 33. 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 $\lbrack \text{Loves}(x, m) \rbrack _{M, g_i} \neq \lbrack \text{Loves}(x, m) \rbrack _{M, g_j}$.

Exercise 34. 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 non-logical constants as you need.
4 Typed lambda calculus

4.1 Introduction

We are still building up the tools that we will need in order to define a system that translates expressions of (a defined fragment of) English into a formal logical language. We will do this in the next chapter. But first, we need to be able to assign meanings to a wider range of expressions. Consider the sentence *Homer loves Maggie*, which might be translated into $L_1$ as:

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

Some parts of the sentence *Homer loves Maggie* can be straightforwardly mapped into expressions in $L_1$, but others do not map onto self-contained chunks. For example, we might say that (relative to a given model) the English name *Maggie* picks out a particular element of the domain, namely Maggie. So it makes sense to translate the English 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 with an empty slot:

$$\text{Loves}(\_, m)$$
where the first argument of Loves is missing. In order to express this idea formally, we need to introduce into our formal system something that can serve as a placeholder.

The language of typed lambda calculus, which we present now, gives us the tools to express this idea of a placeholder. Using the symbol, we can abstract over the missing piece. The result looks like this:

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

This expression (read 'lambda x dot loves x m') denotes a function from an individual to a truth value, which yields true if and only if that individual loves Mary—more or less what loves Mary means.

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

\[ \forall x[\text{___}(x)] \]

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

\[ \forall x[\text{Temporary}(x)] \]

while Everything is permanent would be expressed:

\[ \forall x[\text{Permanent}(x)] \]

In the language that we will consider now, we can have variables ranging over predicates, which can then be abstracted over. The result looks like this:

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

This expression (read ‘lambda P dot for all x: P x’) denotes a function from a predicate \( P \) to a truth value: true if everything satisfies \( P \), and false otherwise. It denotes a function that expects a predicate, and returns a truth value that depends on the input predicate. In this chapter, we will define the syntax and semantics of a language that includes this lambda operator.
4.2 Lambda abstraction

4.2.1 Types

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. The two basic types we will have are:

$$e$$

for individual-denoting expressions, and

$$t$$

for formulas. We will also have complex types such as:

$$\langle e, t \rangle$$

for expression denoting functions from individuals to truth values. The set of types is defined recursively 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.

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.

Note that these types are syntactic categories, not categories of meanings. However, they are associated with categories of meanings. Each type has an associated domain, the set of possible meanings for expressions of that type. We use $D_\tau$ to signify the set of possible denotations for an expression of type $\tau$ (for any type
Typed lambda calculus

(\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; \( D_t = \{1, 0\} \). An expression of type \( \langle e, t \rangle \) denotes a function from individuals to truth values; \( D_{\langle e, t \rangle} \) is the set of functions that take as input an individual, and give a truth value as output. And so forth.

Although the set of types is infinite, there are limits: Not everything is a type. For example, \( \langle e, e \rangle \) is not a type according to this system even though it is sometimes treated as such in the literature; angle brackets are only introduced for complex types according to this definition. Under these definitions, \( \langle e \rangle \) is incorrect; the type of individual-denoting expressions is just bare \( e \).

We also lack types corresponding to sets and binary relations, the sorts of things that the unary and binary predicates of predicate logic denote. In this language, an expression cannot denote a set, because there is no type for that. There is, however, the type \( \langle e, e \rangle \), which corresponds to the characteristic function of a set (a function that takes an individual, and returns true or false depending on whether that individual is in the set). From the characteristic function of a set, one can figure out what the members of the set are, so unary predicates can be replaced by expressions of type \( \langle e, t \rangle \) with no loss of information.

Similarly, an expression cannot denote a binary relation, as there is no type for that. But we do have the type \( \langle e, \langle e, t \rangle \rangle \), which can encode a relation, using a method known as Schönfinkelizing the underlying binary relation (also known as Currying; see Heim & Kratzer (1998) for scholarly notes on these terms). For a given binary relation \( R \), Schönfinkelizing produces a function \( f \) such that \( f(y)(x) = 1 \) if and only if \( \langle x, y \rangle \in R \) (where \( f(y)(x) \) denotes the result of first applying \( f \) to \( y \), and then applying \( f(y) \) to \( x \)).

An example denotation for a type \( \langle e, \langle e, t \rangle \rangle \) expression might be the following function:
Call this function \( f \). Applied to a given argument, say Björn, it gives back another function:

\[
\begin{align*}
\text{Björn} & \\
\rightarrow & \\
\begin{bmatrix}
\text{Agneta} & \rightarrow & 0 \\
\text{Björn} & \rightarrow & 0 \\
\text{Benny} & \rightarrow & 0 \\
\text{Anni-Frid} & \rightarrow & 1 \\
\end{bmatrix}
\end{align*}
\]

The function \( f \) is what results when the following relation is Schönfinkelized:

\[
\{(\text{Björn, Björn}), (\text{Anni-Frid, Björn}), (\text{Björn, Benny}), (\text{Agneta, Anni-Frid})\}
\]

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

As it turns out, Schönfinkelized relations are precisely what we need in order to give a compositional analysis of sentences in natural languages containing transitive verbs. In the next chapter,
we will characterize transitive verbs as denoting such Schönfinke-
lized relations – functions which, when given an individual, return
another function. Thus rather than translating the verb loves as
the binary predicate Loves, we will translate it as a function that
applies to its object (say, Björn, in Agneta loves Björn) to return a
new function, which then may apply to the subject (say, Agneta).
That way, every part of the sentence is compositionally assigned a
meaning, including the verb phrase (loves Björn), and the compo-
sition proceeds in a straightforward manner.

To represent the meaning of the verb loves using a constant of
type $\langle e, \langle e, t \rangle \rangle$, we will use something like

\[
\text{loves}
\]

starting with lower case, in order to show that it denotes a function
rather than a relation. Then

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

will mean that $b$ loves $a$. But as we generally prefer to read the
subject before the object, and in order to reduce parenthesis clut-
ter, we will introduce a notation convention so that instead of
\[
\text{loves} (b) (a)
\]

we will write as a shorthand:

\[
\text{Loves} (a, b)
\]

We will stick to this relational style (as opposed to functional
style) as much as possible. Thus instead of:

\[
\text{loves}
\]

we will represent the meaning of a transitive verb in lambda cal-
culus as:

\[
\lambda y. \lambda x. \text{Loves} (x, y)
\]

This expression denotes the result of Schönfinkelizing the origi-
nal relation denoted by the binary predicate Loves in predicate
logic. Using the relational style helps bring out visually to the reader how many arguments the verb expects to combine with, and is more similar to how verb meanings are represented in current semantics research following the style of [Heim & Kratzer (1998)]; the meaning of the verb *loves* in that style would be represented as ‘\(\lambda x. \lambda y. x \text{ loves } y\)’, with a blend of English and lambda calculus.

By the same token, we define upper-case Happy(\(\alpha\)) to be equivalent to happy(\(\alpha\)), if happy is a predicate of type \(\langle e, t \rangle\). This way, everything that was well-formed in predicate logic will also be well-formed in our language.

### 4.2.2 Syntax and semantics

The introduction of an infinite set of syntactic categories sets the stage for the introduction of the **LAMBDA OPERATOR** (or \(\lambda\)-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:

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

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

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

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. \(\lambda x. \text{Loves}(m, x)\) denotes the characteristic function of the set \(\{x \mid \text{Maggie loves } x\}\).

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 \(\langle e, t \rangle\). More generally:

**Syntax Rule: 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 \(\langle \sigma, \tau \rangle\).
Where \( x \) is a variable over individuals, and \( \phi \) is any formula, a lambda expression of the form \( \lambda x. \phi \) will denote a function from individuals to truth values. In general, the expression

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

denotes a function that returns the value described by \( \phi \) when given input \( x \). Officially, the semantics is defined as follows:

**Semantic Rule: Lambda abstraction**

If \( \alpha \) is an expression of type \( \tau \) and \( u \) a variable of type \( \sigma \) then \( [\lambda u. \alpha]^Mg \) is that function \( h \) from \( D_{\sigma} \) into \( D_{\tau} \) such that for all objects \( k \) in \( D_{\sigma} \), \( h(k) = [\alpha]^Mg[u \mapsto k] \).

For example, \( \lambda x. \mathrm{Happy}(x) \) is of the form \( \lambda u. \alpha \) where \( u \) (= \( x \)) is of type \( e \), and \( \alpha \) (= \( \mathrm{Happy}(x) \)) is of type \( t \). So it denotes the function \( h \) from \( D_e \) to \( D_t \) such that for all objects \( k \) in \( D_e \), \( h(k) \) is equal to \( [\mathrm{Happy}(x)]^Mg[u \mapsto k] \). For any object \( k \), \( h(k) \) will return ‘true’ if \( k \) is happy, and ‘false’ if not. So \( \lambda x. \mathrm{Happy}(x) \) denotes the characteristic function of the set of happy people. If this seems overwhelming, stay calm; it may start to sink in after you get some practice with beta reduction, which we turn to next.

### 4.2.3 Application and beta-reduction

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, \( [\lambda x. \mathrm{Happy}(x)](a) \) denotes the result of applying the function denoted by \( \lambda x. \mathrm{Happy}(x) \) to the semantic value of \( a \). This principle also applies to syntactically complex function-denoting terms.
formed by lambda abstraction. Thus

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

denotes the result of applying the function ‘loves Björn to Agneta.

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

\[ \text{Loves}(ag, bj) \]

where the \( \lambda \)-binder and the variable have been removed, and we have kept just the part after the dot, with the modification that the argument of the function is substituted for all instances of the variable. This kind of simplification is known as \( \beta \)-REDUCTION or \( \beta \)-CONVERSION.

Typically, in a lambda expression of the form \( \lambda x. \phi \), the \( \phi \) part will involve \( x \) (although the function might just ignore its input entirely and return the same thing regardless of what input it gets; that would be a case of VACUOUS ABSTRACTION). Because the part that comes after the dot describes the value of the function given an argument, it is called the VALUE DESCRIPTION. For example, the value description in the expression

\[ \lambda x. \text{Loves}(x, bj) \]

is

\[ \text{Loves}(x, bj) \]

**Exercise 1.** Identify the value description in the following lambda expressions:

1. \( \lambda x. \text{Happy}(x) \)
2. \( \lambda x. x \)
3. \( \lambda y. \lambda x. [\text{Loves}(x, y) \lor \text{Loves}(y, x)] \)
4. \( \lambda z. \lambda y. \lambda x. \text{Between}(x, y, z) \)
The following pairs of expressions, where the second is a beta-reduced version of the other, are equivalent.

1. a. \( [\lambda x. \text{Smiled}(x)](a) \)
   b. \( \text{Smiled}(a) \)
2. a. \( [\lambda x. [\text{Smiled}(x) \land \text{Happy}(x)]](a) \)
   b. \( [\text{Smiled}(a) \land \text{Happy}(a)] \)
3. a. \( [\lambda x. [\text{Smiled}(x) \land \text{Happy}(y)]](a) \)
   b. \( [\text{Smiled}(a) \land \text{Happy}(y)] \)

In general, the result of applying the function to the argument can be described more simply by taking the value description and replacing all free occurrences of the lambda-bound variable with the argument.

We say “all free occurrences” because if another variable binder such as a universal or an existential quantifier is present in the formula, then occurrences of the lambda-bound variable may in principle be bound by that operator, and then the variable is no longer bound by the lambda operator. Thus the following pairs of formulas are equivalent:

4. a. \( [\lambda x. [\text{Smiled}(x) \land \exists x. \text{Happy}(x)]](a) \)
   b. \( [\text{Smiled}(a) \land \exists x. \text{Happy}(x)] \)

The instances of the variable \( x \) that are inside the scope of the existential quantifier are bound by that quantifier, so replacing them with \( a \) will not result in an equivalent expression.

There is one other nitpicky issue we have to address before moving on from the topic of beta reduction, and that is what happens if the argument itself is a variable, or contains a free variable. For example, consider:

5. \( [\lambda y. \lambda x. \text{Loves}(x, y)](x) \)

If we just substitute \( x \) in for \( y \), we get:
(6) \( \lambda x. \text{Loves}(x, x) \)

But this is not equivalent to the original expression. Through overly enthusiastic substitution of \( y \) for \( x \), the variable \( y \) accidentally became bound by the inner lambda operator.

Observe that the inner lambda expression could have involved any variable. It didn’t have to be \( x \). For example, it could have been \( z \):

(7) \( [\lambda y. \lambda z. \text{Loves}(z, y)](x) \)

and this would have had exactly the same meaning as (5). Generally, the principle of ALPHA EQUIVALENCE holds for lambda calculus as we have defined it: a bound variable can be replaced by any other variable without a change in meaning. Notice that if we substitute \( x \) into the version with \( z \) in (7) then we get a very different result:

(8) \( \lambda z. \text{Loves}(z, x) \)

Whereas our former attempt in (6) expresses the property of loving oneself, this expresses the property of loving whomever \( x \) picks out.

In fact, beta-reduction exercises with the Lambda Calculator require students to first take advantage of the principle of alpha equivalence when doing beta reduction on arguments that contain free variables that occur in the value description. The first step when reducing an expression like (5) is to ‘re-letter’ the bound variable as in (7) and then carry out beta-reduction as usual.

We are now in a position to state the rule of beta reduction in its full glory:

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

where \( \phi[x := \alpha] \) stands for the result of replacing all free occurrences of \( x \) with \( \alpha \) in \( \phi \), as long as \( \alpha \) does contain any free variables that occur in \( \phi \). Note that \( \beta \)-reduction does not need to be
stipulated. It’s just a fact that follows from how the semantics of $\lambda$-abstraction is defined.

### 4.2.4 Some applications

Our new and improved representation language, with its capacity for abstraction and its infinitely many types, can express a wide range of potential 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 `smoker` denotes the set of people who smoke:

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

Then `non-smoker` should be the set of people who don’t smoke:

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

This expression denotes a function which takes an individual and returns 1 if and only if that individual is not a smoker.

Somewhat informally speaking, a `non-P` is someone who satisfies $\lambda x. \neg P(x)$. So the meaning of `non-` can be represented as a function that takes as its argument a predicate (say, $P$) and then returns a new predicate which holds of an individual if the individual does not satisfy the input predicate $P$:

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

If we apply this function to $\lambda x. \text{Smokes}(x)$, the result is equivalent to $\lambda x. \neg \text{Smokes}(x)$, which correctly captures the fact that a `non-smoker` doesn’t smoke. As the prefix `non-` applies to a predicate, rather than an individual, it can be said to denote a higher-order function, that is, a function that applies to other functions. By the same token, `non-` is a higher-order expression.

Other higher order expressions that we can represent with our new language include `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 $P$ and $Q$ be variables ranging over the characteristic functions of sets (type $(e, t)$). The meaning of _every_ can be represented like this:

\[
\lambda P. [\lambda Q. \forall x. [P(x) \to Q(x)]]
\]

This expression denotes a function which takes a predicate (call it $P$), and returns a function that takes another predicate (call it $Q$), and returns $1$ (true) if and only if every $P$ is a $Q$.

The denotation of _every cellist_ would be the result of applying this function to the denotation of _cellist_. This means that _cellist_ must be denote a function from individuals to truth values. This can be achieved if we stipulate that _cellist_ is translated as follows:

\[
\lambda y. \text{Cellist}(y)
\]

Then _every cellist_ will be translated as:

\[
[\lambda P. [\lambda Q. \forall x. [P(x) \to Q(x)]]][\lambda y. \text{Cellist}(y)]
\]

The translation at the top node beta-reduces to:

\[
\lambda Q. \forall x. [[\lambda y. \text{Cellist}(y)](x) \to Q(x)]
\]

which in turn beta-reduces to:

\[
\lambda Q. \forall x. \text{Cellist}(x) \to Q(x)
\]

Thus the meaning of _every_, applied to the meaning of _cellist_, is a function that is still hungry for another unary predicate. Feeding it $\lambda z. \text{Smokes}(z)$ produces a sentence that denotes a truth value:

\[
[\lambda Q. \forall x. [\text{Cellist}(x) \to Q(x)]]([\lambda z. \text{Smokes}(z)])
\]
\[ \equiv \forall x. [\text{Cellist}(x) \rightarrow \text{Smokes}(x)] \]

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 2.** 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.

### 4.3 Summary

#### 4.3.1 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:

- \( e \) 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.

A **Formula** is an expression of type \( t \).

Under the hood, we will simply assume that there are an infinite set of constants and variables of every type. Outside of the official syntax, we will allow ourselves to define particular constants like \texttt{Happy} and \texttt{motherOf} to be equivalent to one of the actual constants in the language.
1. **Basic Expressions**
   For every natural number $n$ and every type $\tau$, there is:
   
   - a constant of the form $c_{\tau,n}$
   - a variable of the form $\nu_{\tau,n}$

2. **Application** (cf. ‘Complex terms’ – 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 $\sigma$ 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 $\neg \phi, [\phi \land \psi], [\phi \lor \psi], [\phi \to \psi]$, 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. Similarly to how we can drop the dot between two quantificational binders, we can also drop the dot between two lambdas in a row, so we can write, e.g. $\lambda x \lambda y . \text{Admires}(x, y)$. We will, however, always retain the final dot in a sequence of lambda binders in
order to show that the end of the argument list has been reached, e.g. \( \lambda x \lambda y . \exists z . \text{Gave}(x, y, z) \). Once again, these dot-related conventions are specific to our textbook, and there is no single standard in the field.

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

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

\[
\text{Loves}(x, y)
\]

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

\[
\pi(\alpha_1)(\alpha_2)...(\alpha_n)
\]

can be re-written as

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

where \( \Pi \) is a variant of \( \pi \) that starts with a capital letter.

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

- \( x \) is \( v_{0,e} \) (meaning that \( x \) is variable number 0 of type \( e \))
- \( y \) is \( v_{1,e} \)
- \( P \) is \( v_{0,(e,t)} \), \( Q \) is \( v_{1,(e,t)} \), and \( X \) is \( v_{2,(e,t)} \)
- \( R \) is \( v_{0,(e \times e,t)} \)
- \( a \) is \( c_{0,e} \) and \( b \) is \( c_{1,e} \)
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 4. Identify the type of each of the following. Assume that \text{man} and \text{mortal} are constants of type \((e, t)\).

1. \(\lambda y \cdot y\)
2. \(\lambda x \cdot P(x)\)
3. \(P\)
4. \(a\)
5. \(x\)
6. \(P(x)\)
7. \([\lambda x \cdot P(x)](a)\)
8. \(P(a)\)
9. \(R(x, y)\)
10. \(\lambda x \cdot R(x, a)\)
11. \(\lambda y \lambda x \cdot R(y, x)\)
12. \([\lambda y \lambda x \cdot R(y, x)](a)\)
13. \([\lambda x \cdot R(y, a)](b)\)
14. \(R(a, b)\)
15. \(\lambda x \cdot [P(x) \land Q(x)]\)
16. \([\lambda x \cdot P(x) \land Q(x)](a)\)
17. \(\lambda x \lambda y \cdot [R(y)(a) \land Q(x)]\)
18. \(\lambda P \cdot P\)
19. \(\lambda P \cdot P(a)\)

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

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

32. $\neg \text{mortal}(a)$

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

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

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

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

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

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

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

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

You can check your answers using the [Lambda Calculator](#).
Exercise 5. Where possible, apply \(\beta\)-reduction to give a more concise version of each of the following. If the expression is fully reduced, just give the original expression.

1. \([\lambda x. x](a)\)
2. \([\lambda P. P](\text{man})\)
3. \([\lambda x. P(x)](a)\)
4. \([\lambda x. P(x)]\)
5. \([\lambda y \lambda x. R(y, x)](a)\)
6. \([\lambda x. R(y, a)](b)\)
7. \([\lambda P. \exists x. P(x)](\text{man})\)
8. \([\lambda P. \forall x. P(x)](\text{mortal})\)
9. \(\lambda x. \neg\text{mortal}(x)\)
10. \([\lambda P \lambda x. \neg P(x)](\text{mortal})\)
11. \([\lambda x. \neg\text{mortal}(x)](a)\)
12. \([\lambda Q. \forall x. [\text{man}(x) \rightarrow Q(x)]](\text{mortal})\)
13. \([\lambda P \lambda Q. \forall x. [P(x) \rightarrow Q(x)]](\text{man})\)
14. \([\lambda P \lambda Q. \forall x. [P(x) \rightarrow Q(x)]](\text{man})(\text{mortal})\)
15. \([\lambda x. P(x) \land Q(x)](a)\)
16. \([\lambda x \lambda y. [R(y, a) \land Q(x)]](a)(b)\)
17. \([\lambda x. \exists y. R(x, y)](y)\)
18. \([\lambda x. a](b)\)
4.3.2 Semantics

As in L₁, the semantic values of expressions in Lₜ depend on a model and an assignment function. As in L₁, 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 τ, I assigns an object of type τ to every non-logical constant of type τ.

Recall that types are associated with domains:

- The domain of individuals $D_e = D$ 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 $σ$ and $τ$, $D_{(σ,τ)}$ is the domain of functions from $D_σ$ to $D_τ$.

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

The semantic value of an expression is defined as follows:

1. **Basic Expressions**
Typed lambda calculus

(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. Application
If $\alpha$ is an expression of type $(\sigma, \tau)$, and $\beta$ is an expression of type $\sigma$, then $\alpha(\beta)[M,g] = \alpha[M,g](\beta[M,g])$.

3. Equality
If $\alpha$ and $\beta$ are expressions of the same type, then $\alpha = \beta \iff \alpha[M,g] = \beta[M,g]$.

4. Negation
If $\phi$ is a formula, then $\neg \phi[M,g] = 1 \iff \phi[M,g] = 0$.

5. Binary Connectives
If $\phi$ and $\psi$ are formulas, then:
(a) $\phi \land \psi[M,g] = 1 \iff \phi[M,g] = 1$ and $\psi[M,g] = 1$.
(b) $\phi \lor \psi[M,g] = 1 \iff \phi[M,g] = 1$ or $\psi[M,g] = 1$.
(c) $\phi \rightarrow \psi[M,g] = 1 \iff \phi[M,g] = 0$ or $\psi[M,g] = 1$.
(d) $\phi \leftrightarrow \psi[M,g] = 1 \iff \phi[M,g] = \psi[M,g]$.

6. Quantification
(a) If $\phi$ is a formula and $\nu$ is a variable of type $\tau$ then $\forall \nu . \phi[M,g] = 1 \iff$ for all $k \in D_\tau$:

$$\phi[M,g][\nu \mapsto k] = 1$$

(b) If $\phi$ is a formula and $\nu$ is a variable of type $\tau$ then $\exists \nu . \phi[M,g] = 1 \iff$ there is an individual $k \in D_\tau$ such that:

$$\phi[M,g][\nu \mapsto k] = 1$$

7. Lambda Abstraction
If $\alpha$ is an expression of type $\tau$ and $u$ a variable of type $\sigma$ then $\lambda u . \alpha[M,g]$ is that function $h$ from $D_\sigma$ into $D_\tau$ such that for all objects $k$ in $D_\sigma$, $h(k) = \alpha[M,g][u \mapsto k]$. 
Exercise 6.

(a) Partially define a model for $L_\lambda$ giving denotations to the constants $\text{loves}$, $n$, and $d$ of type $(e,(e,t))$, $e$, and $e$, respectively.

(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 7. Relational kinship terms like $\text{aunt}$ can be thought of as denoting binary relations among individuals. We might therefore introduce a binary predicate $\text{Aunt}$ to represent the aunthood relation, such that a sentence like $\text{Sue is Alex's aunt}$ could be represented as $\text{Aunt}(\text{sue, alex})$. But consider $\text{Sue is an aunt now!}$ This sentence might be taken to express an existential claim like $\exists x.\text{Aunt}(\text{sue, x})$. On such a usage, the noun $\text{aunt}$ 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 $\text{aunt}$ 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 $(e,(e,t))$) 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 $(\langle e,\langle e, t \rangle \rangle,\langle e, t \rangle)$. 
**Exercise 8.** We normally consider *eat* a transitive verb, and according to the kind of analysis we have done here, this would imply a treatment as a binary relation, type \( \langle e, \langle e, t \rangle \rangle \). And yet we do have usages where the object does not appear, as in *Have you eaten?* One might imagine that a two-place predicate can be reduced to a one-place predicate through an operation that existentially quantifies over the object argument. Define a function that does this and express it as a well-formed lambda term in \( L_\lambda \). 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 9.** Like *eat*, the verb *shave* can be used both transitively and intransitively; consider *The barber shaved John* and *The barber shaved*. But in contrast to *eat*, the intransitive version does not mean that the barber shaved something; it means that the barber shaved himself. Give an expression of \( L_\lambda \) 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.
5  |  Function application

5.1 Introduction

Recall Frege's conjecture: *All composition is function application.* In this chapter, we will define a semantics for a fragment of English that adheres to this principle. To do so, we will translate expressions of English into expressions of $\lambda$. A name like Agneta will translate as the type $e$ expression $ag$, and the intransitive verb will be translated as the type $\langle e, t \rangle$ expression $\lambda x. \text{Smiled}(x)$:

- $Agneta \sim ag$
- $\text{smiled} \sim \lambda x. \text{Smiled}(x)$

where $\sim$ signifies the ‘translates as’ relation. The combination, *Agneta smiled*, will then be translated as the result of applying the translation of the verb to the translation of the subject:

\[
\left[ \lambda x. \text{Smiled}(x) \right](ag)
\]

... or equivalently, through beta reduction:
Again, note that we are using an indirect interpretation method in this book, which means that we translate English to the representation language first (using \(~\)\), and then interpret the representation language (using \([\cdot]\))\). So rather than the Heim & Kratzer (1998) style:

\[[Agneta smiled] = 1\]

we instead write:

\(Agneta smiled \sim Smiled(\text{ag})\)

and:

\([\text{Smiled(\text{ag})}] = 1\)

We take the semantics of the English expressions be inherited from their translations in lambda calculus\(^1\)\). A given sentence can then be said to be true with respect to a model and an assignment function if its translation is true with respect to that model and assignment function.

Indirect translation is the style that Montague (1974b) used 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

---

\(^1\)Assuming that there may be multiple translations into the representation language for a given expression of English, there is not necessarily a unique semantic value, so the English-to-meaning mapping would be one-to-many in principle, under our assumptions.
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).\(^2\)

A fully specified fragment of English consists of:

- A specification of our formal representation language, with syntactic and semantic rules
- A specification of the syntax of the English expressions we cover
- A list of lexical entries
- A list of composition rules

In ‘The Proper Treatment of Quantification in Ordinary English’, Richard Montague formally defined the first fragment of English. Throughout this book we too will build up a fragment that is comparable to the one Montague developed.

We already have our representation language: \(L_\lambda\). The next step is to specifying the rules that generate the syntactically well-formed expressions of our fragment of English.\(^3\)

(1) **Syntax**

\(^2\)An important difference between the tact we are taking here and the one taken in Heim & Kratzer’s 1998 textbook is that here the \(\lambda\) symbol is used as an expression of lambda calculus (its original use), whereas in Heim and Kratzer the \(\lambda\) symbol is part of the meta-language, as an abbreviation for describing functions. One should carefully distinguish between these two ways of using it.

\(^3\)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 *Agneta* are of the same category, this means that *Agneta* is also a DP. We assume therefore that proper
Function application

\begin{align*}
S & \rightarrow \text{DP VP} \\
S & \rightarrow \text{Neg S} \\
S & \rightarrow \text{S JP} \\
\text{JP} & \rightarrow \text{J 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*}

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:

\((2)\) **Lexicon**

- **J**: and, or
- **Neg**: it-is-not-the-case-that
- **V**: smiled, laughed, loves, hugged, is
- **A**: Swedish, happy, kind, proud
- **N**: singer, drummer, musician
- **D**: the, a
- **D**: Agneta, Anni-Frid, Björn, Benny
- **P**: of, with

For example, this grammar generates *Björn is the drummer* and *it-is-not-the-case-that Benny smiled*, with syntactic structures as shown in the following analysis trees:

---

names are of category D, like articles. The difference is that proper names are 'intransitive', so they don't combine with an NP.
**Exercise 1.** Which of the following strings are sentences of the fragment of English that we have defined (modulo sentence-initial capitalization)? Draw syntax trees for those that are.

(a) George loves everybody.

(b) Some drummer smiled every happy musician.

(c) Agneta is not a drummer.

(d) Anni-Frid is.

(e) No is a happy singer.

(f) Somebody is proud of the singer.

(g) A drummer loves proud of Björn.

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

(i) Anni-Frid smiles with nobody.

(j) Agneta and Anni-Frid are with Björn.
(k) Agneta is with Björn and Anni-Frid is with Benny.

Keep in mind that the syntax might generate sentences that don’t make any sense, and that’s OK. At least some of the nonsensical sentences will be ruled out once we define semantic interpretations for these words.

Note: In the trees below, sometimes we “prune” non-branching nodes. For example, we might write:

```
   DP
     |
    Agneta
```

instead of

```
   DP
     |
     D
     |
    Agneta
```

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.

In accordance with Frege’s conjecture, we have only one rule for composing the meaning of a complex expression out of the meanings of the parts, namely FUNCTION APPLICATION, which just applies a function to an argument:
**Composition Rule 1. Function Application** (FA)
Let $\gamma$ be a tree whose only two subtrees are $\alpha$ and $\beta$ where:

- $\alpha \sim \alpha'$ where $\alpha'$ has type $\langle \sigma, \tau \rangle$
- $\beta \sim \beta'$ where $\beta'$ has type $\sigma$.

Then

$$\gamma \sim \alpha'(\beta')$$

(Note: The prime symbol $'$ in $\alpha'$ is not intended to have any meaning of its own; $\alpha'$ is just a convenient way to refer to whatever $\alpha$ is translated as. We could have referred to it as $A$ or $XYX$ or *Snuf-falupagus* instead; we just needed to call it something.)

This rule will provide a translation into $L_\lambda$ for any tree that has two immediate subtrees, as long as their types match appropriately. The node at the top of such a tree is called a **branching node** because it branches into multiple subtrees. If a tree has no branches, then it is called a **non-branching node**. For non-branching nodes, we will simply assume that the denotation of the larger tree is the same as the denotation of the single subtree:

**Composition Rule 2. Non-branching Nodes** (NN)
If $\beta$ is a tree whose only daughter is $\alpha$, where $\alpha \sim \alpha'$, then $\beta \sim \alpha'$.

With these two rules, we can assign denotations to each subtree in the syntactic structure of *Agneta smiled* as follows (showing only fully beta-reduced translations at each node, along with their types):
5.2 Fun with Function Application

5.2.1 Agneta loves Björn

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

(3)  \[\text{loves} \rightarrow \lambda y \lambda x. \text{Loves}(x, y)\]

Can this verb combine semantically with a type-\(e\) direct object via Function Application? Yes, it can; the types match. This is shown in the following derivation for *Agneta loves Björn*:
Via Function Application, the transitive verb *loves* combines with the object *Björn*. The VP *loves Björn* thus comes to denote the property of loving Björn. This property is then attributed to Agneta through a second application of Function Application at the top node.

**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.

(a)
5.2.2 Homer is lazy

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

We will continue to assume that the proper name Björn is translated as the constant bj, of type e. We can assume that kind denotes a function of type \(\langle e, t \rangle\), the characteristic function of a set of individuals (those that are kind). Let us use Kind as an abbreviation for a constant of type \(\langle e, t \rangle\), and translate kind thus.

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

Now, what is the contribution of is? Besides signaling present tense, it does not seem to accomplish more than to link the predicate ‘kind’ with the subject of the sentence. Since we have not starting dealing with tense yet, we will ignore the former function and focus on the latter. We can capture the fact that is connects
the predicate to the subject by treating it as an identity function, a function that returns whatever it takes in as input. In this case, \( \text{is} \) takes in a function of type \(<e, t>\), and returns that same function.

(5) \( \text{is} \rightarrow \lambda P. P \)

This implies that \text{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 \textit{Björn is kind}:

(6)

Each node shows the syntactic category, the semantic type, and a fully \( \beta \)-reduced translation to lambda calculus. In this case, 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 (\textit{Björn}, \textit{is}, and \textit{kind}).
5.2.3 Björn is not kind

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

![Syntax Tree](image)

The denotation of *Björn is kind* should be the negation of *Björn is kind*:

\[ \neg \text{Kind}(bj) \]

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

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

The meaning of *not* should apply to a property and produce such a function for any arbitrary predicate, not just *kind*. The following meaning will do the trick:

\[ (7) \quad \text{not} \leadsto \lambda P \lambda x. \neg P(x) \]

**Exercise 3.** Using this lexical entry for *not*, give a compositional analysis of *Björn is not kind*, by showing the translations and types at each node of the syntax tree.
5.2.4  Anni-Frid is with Benny

Like adjectives, prepositional phrases can also serve as predicates, as in, for example, *Anni-Frid is with Benny*. Let us translate *with* as follows, invoking a binary predicate $\text{With}$:

\[
(8) \quad \text{with} \sim \lambda y \lambda x. \text{With}(x, y)
\]

Here is an overview of how the derivation will go. Via Functional Application, the preposition *with* combines with its object *Benny*, and the resulting PP combines with *is* to form a VP. The translation of the VP is an expression of type $\langle e, t \rangle$, denoting a function from individuals to truth values. This applies to the denotation of *Anni-Frid* to produce a truth value.

Exercise 4. Derive the translation into $L_\lambda$ for *Anni-Frid is with Benny* by giving a fully $\beta$-reduced translation for each node.
5.2.5  **Benny is proud of Anni-Frid**

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

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

(9)  \( \text{of} \sim \lambda x . x \)

So the adjective phrase *proud of Anni-Frid* will have the following structure:

Exercise 5. Give a lexical entry for *proud* and a fully \( \beta \)-reduced form of the translation at each node for *Benny is proud of Anni-Frid*. (You will need to draw out more of the tree structure than
5.2.6 Agneta is a singer

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

The indefinite article a is another function word that appears to be semantically vacuous, at least on its use in the present context. We will assume that a, like is, denotes a function that takes an \( \langle e, t \rangle \)-type predicate and returns it.

(10) \[ a \sim \lambda P.P \]

With these assumptions, the derivation will go as follows.

(11) 

```
S
 t

  DP
  e

 Agneta

  VP
  \langle e, t \rangle

  V
  \langle \langle e, t \rangle, \langle e, t \rangle \rangle

  is

  DP
  \langle e, t \rangle

  NP
  \langle \langle e, t \rangle, \langle e, t \rangle \rangle

  a

  singer
```
Exercise 6. Give fully $\beta$-reduced translations at each node of the tree for *Agneta is a singer*.

Exercise 7. Can we treat $a$ as $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ in a sentence like *A singer loves Björm*? Why or why not?

Exercise 8. Assume that *Norwegian* and *millionaire* are both of type $\langle e,t\rangle$, following the style we have developed so far. Is it possible to assign truth conditions to the following sentence using those assumptions? Why or why not?

5.3 Quantifiers: A sneak peek

Now let us briefly consider how to analyze quantifiers like *every-body* and *nobody*, a topic we will return to in more depth in Chapter 7. Consider the sentence:
(12) Everybody smiled.

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

(13) \(\textit{everybody} \text{ smiled} \sim \forall x.\text{Smiled}(x)\)

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

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

where the verb phrase fills in the underlined slot. This idea can be formally implemented through lambda abstraction. Everybody will denote a function that takes an arbitrary predicate \(P\), and yields a truth value: true if everything satisfies \(P\) and false if not:

(14) \(\textit{everybody} \sim \lambda P. \forall x. P(x)\)

As \(P\) is a variable that stands for a predicate—something of type \(\langle e, t \rangle\)—the type of the expression denoted by everybody is:

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

This is the type of a QUANTIFIER.

This meaning for everybody can be combined via Function Application with the meaning for smiled in the following manner:

\[ \forall x.\text{Smiled}\langle x \rangle \]

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

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

\[ \langle e, t \rangle \]

\[ \textit{everybody} \]

\[ \textit{smiled} \]
In this derivation, the VP is fed as an argument to the subject DP. Recall that Function Application does not care about the order of the arguments, so this order of application works just as well as the more familiar situation where the VP takes the subject as an argument.

We can define something and nothing similarly:

\[
\begin{align*}
(15) \quad & \text{something} \sim \lambda P \cdot \exists x. P(x) \\
(16) \quad & \text{nothing} \sim \lambda P \cdot \neg \exists x. P(x)
\end{align*}
\]

All of these quantifiers can be thought of as predicates of predicates. For example, \( \lambda P \cdot \neg \exists x. P(x) \) denotes a predicate that holds of a predicate over individuals if it has no satisfiers.

So far, so good. But consider what happens if the quantifier is in object position, as in, for example Björn loves everyone? It is clear what logical formula would serve to represent the meaning of this sentence:

\[
(17) \quad \text{Björn loves everyone} \sim \forall x. \text{Loves}(bj, x)
\]

But it is entirely unclear, given what we have said so far, how this meaning could be derived compositionally, given that neither the verb nor the object can take the other as an argument. The transitive verb loves is expecting an argument of type \( e \), but the quantifier is not of that type. Conversely, the quantifier is expecting an argument of type \( (e, t) \), but the verb is not of that type. Hence, we have a type clash:
This predicament is known as the **PROBLEM OF QUANTIFIERS IN OBJECT POSITION**. We will offer a resolution to this problem in Chapter 7 after developing the necessary compositional machinery in Chapter ???. Specifically, we will introduce the rule of **PREDICATE ABSTRACTION** in order to deal with relative clauses, and this will provide a mechanism for solving the problem of quantifiers in object position, in conjunction with a covert transformation known as **QUANTIFIER RAISING**.

### Exercises

**Exercise 9.** 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 10. In some languages, there is a morpheme (e.g., Middle Voice in Ancient Greek, reflexivizing affix in Kannada, Passive Voice in Finnish, etc.) that attaches to the verb stem and reduces its arity by one. Let us take the following imaginary morphemes `self1`, `self2`, and `self3`. Assuming the syntactic structure given, give a denotation for each of these morphemes.

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

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

![Diagram](https://via.placeholder.com/150)
(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*.

```
 Carlos
    /     \
   self3   introduced
     \     /    /
          Paco
```

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

(Exercise due to Maribel Romero.)

### 5.3.1 Toy fragment

So far, we have developed the following toy fragment of English, consisting of a set of syntax rules, a lexicon, a set of composition rules, and a set of lexical entries.

**Syntax**

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

**Lexicon**
J: and, or
Neg: it-is-not-the-case-that
V: smiled, laughed, loves, hugged, is
A: Swedish, happy, kind, proud
N: singer, drummer, musician
D: the, a
D: Agneta, Anni-Frid, Björn, Benny
P: of, with

Composition Rules

- **Functional Application** (FA)
  Let $\gamma$ be a tree whose only two subtrees are $\alpha$ and $\beta$ where:
  
  - $\alpha \sim \alpha'$ where $\alpha'$ has type $\langle \sigma, \tau \rangle$
  - $\beta \sim \beta'$ where $\beta'$ has type $\sigma$.

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

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

Lexical entries

- $Agneta \sim ag$
- $smiled \sim \lambda x. Smiled(x)$
- $loves \sim \lambda y \lambda x. Loves(x, y)$
- $kind \sim \lambda x. Kind(x)$
- $is \sim \lambda P. P$
- $with \sim \lambda y \lambda x. With(x, y)$
• \textit{of} \sim \lambda x. x

• \textit{a} \sim \lambda P. P

• \textit{not} \sim \lambda P \lambda x. \neg P(x)

\textbf{Exercise 11.} Extend this fragment to assign representations in $L_\lambda$ to the following sentences so that the following two sentences are predicted to be equivalent. For both sentences, give a parse tree with a fully beta-reduced representation at each node.

• \textit{Björn is a fan of Agneta.}

• \textit{Björn is Agneta’s fan.}

\textbf{Exercise 12.} Extend the fragment to assign representations in $L_\lambda$ to the following sentences so that the following two sentences are equivalent. For both sentences, give a parse tree with a fully beta-reduced representation at each node.

• \textit{Björn smokes and Agneta drinks.}

• \textit{Benny smokes and drinks.}
6 Other composition rules

6.1 Predicate Modification

6.1.1 Anni-Frid is a Norwegian millionaire

In the previous chapter, we considered sentences like Anni-Frid is Norwegian and Anni-Frid is a millionaire, but we did not have the tools to analyze sentences like:

(1) Anni-Frid is a Norwegian millionaire.

According to our lexical entry for Norwegian above, Norwegian denotes a function of type $\langle e, t \rangle$, and so does millionaire. 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 Functional Application cannot be used to combine them, and so far, we have no other rules that could be of use.

One type of strategy we could pursue in order to rectify this situation is to posit a meaning for Norwegian that can be applied
directly to *millionaire*. The shifted meaning would expect a predicate like millionaire, and return a new predicate that holds of individuals that are in the intersection between Norwegians and the input predicate:

\[(2)\quad \lambda P. \lambda x. [\text{Norwegian}(x) \land P(x)]\]

This expression, which is of type \(\langle (e, t), (e, t) \rangle\), could be derived from the basic \(\langle e, t \rangle\) meaning for *Norwegian* with the help of a TYPE-SHIFTING RULE. The type-shifting rule that we would need in this case would introduce a translation of type \(\langle (e, t), (e, t) \rangle\) for every translation of type \(\langle e, t \rangle\) (Partee, 1995, p. 29).

There are two technical ways we could implement type-shifting. One would be to specify that, in addition to the translation for *Norwegian* of type \(\langle e, t \rangle\) that we have already defined, there is in addition a second translation of type \(\langle (e, t), (e, t) \rangle\). Another way to do it would be to assume that type-shifters are present in the syntax tree. So the type-shifted version of *Norwegian* will technically be like this syntactically:

\[(3)\quad \begin{align*}
A & \\
\langle (e, t), (e, t) \rangle & \\
\text{MOD} & \\
A & \\
\langle e, t \rangle &
\end{align*}
\]

We will remain neutral between these two approaches, but we use a special notation for derivations involving type-shifting operations, using the upwards-facing double arrow \(\uparrow\) in order to capture the intuition that the type-shifting operation induces a transformation of the meaning:
Other composition rules

The MOD rule is defined as follows:

**Type-Shifting Rule 1.** Predicate-to-modifier shift (MOD)
If $\alpha \sim \alpha'$, where $\alpha'$ is of type $\langle e, t \rangle$,

\[
\text{then } \alpha \sim \lambda P \cdot [\alpha'(x) \land P(x)] \text{ (as long as } P \text{ and } x \text{ are not in } \alpha'; \text{ in that case, use different variables of the same type).}
\]

Thus *Norwegian millionaire* gets the following analysis:

\[
\begin{align*}
\text{NP} & \quad \lambda x. [\text{Norwegian}(x) \land \text{Millionaire}(x)] \\
\quad & \quad \lambda P \lambda x. [\text{Norwegian}(x) \land P(x)] \quad \lambda x. \text{Millionaire}(x) \\
\quad & \quad \lambda x. \text{Norwegian}(x) \\
\quad & \quad \text{Norwegian}
\end{align*}
\]

This type-shifting solution is similar to Montague's strategy for dealing with adjectives that modify nouns. In fact, *all* adjectives are $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$ for Montague. For a sentence like *Anni-Frid is Norwegian*, 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 *Anni-Frid is Norwegian millionaire* 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 \(\langle e, t \rangle\).

**Exercise 1.** There is another possible type-shifting analysis, where the version with the more complicated type \(\langle\langle e, t \rangle, \langle e, t \rangle\rangle\) is basic, and the predicative, \(\langle e, t \rangle\) version is derived through a type-shifting rule. Specify an appropriate such rule.

Instead of using a type-shifting rule to interpret attributive adjectives, another strategy is to let go of Frege’s conjecture (that Function 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:

- \(\gamma\) is a tree whose only two subtrees are \(\alpha\) and \(\beta\)
- \(\alpha \sim \alpha'\)
- \(\beta \sim \beta'\)
- \(\alpha'\) and \(\beta'\) are of type \(\langle e, t \rangle\)

Then:

\[
\gamma \sim \lambda u. [\alpha'(u) \land \beta'(u)]
\]

where \(u\) is a variable of type \(e\) that does not occur free in \(\alpha'\) or \(\beta'\).
This gives us the following derivation for the NP:

(6) \[
\begin{array}{c}
\text{NP} \\
\langle e, t \rangle \\
\lambda x. [\text{Norwegian}(x) \land \text{Millionaire}(x)] \\
\end{array}
\]

A

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

\[
\begin{array}{c}
\langle e, t \rangle \\
\lambda x. \text{Millionaire}(x) \\
\text{millionaire} \\
\end{array}
\]

**Exercise 2.** Consider the sentence *Maggie is a Norwegian 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 $\beta$-reduced translation. (Feel free to use the Lambda Calculator for this.)

**Exercise 3.** Identify the types of the following expressions:

(a) $\lambda x \lambda y. \ln(y, x)$

(b) $\lambda x. x$

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

(d) texas

(e) $\lambda y. \ln(y, \text{texas})$

(f) $\lambda f. f$

(g) $\lambda y \lambda x. \text{Fond-of}(x, y)$
Assume:

- $x$ and $y$ are variables of type $e$, and $f$ is a variable of type $\langle e, t \rangle$.
- Any constant that appears with an argument list of length 1 (e.g. City) is a unary predicate, and any constant that appears with an argument list of length 2 (e.g. In).
- Any constant that appears without an argument list (e.g. 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 4.** In addition to the ones given above, adopt the following lexical entries, using the same assumptions about types as in the previous exercise:

1. $cat \sim \lambda x. \text{Cat}(x)$
2. $city \sim \lambda x. \text{City}(x)$
3. $gray \sim \lambda x. \text{Gray}(x)$
4. $gray_2 \sim \lambda P \lambda x. \text{Gray}(x) \land P(x)$
5. $in \sim \lambda y \lambda x. \text{In}(x, y)$
6. $in_2 \sim \lambda y \lambda P \lambda x. P(x) \land \text{In}(x, y)$
7. $fond \sim \lambda y \lambda x. \text{FondOf}(x, y)$
8. $fond_2 \sim \lambda y \lambda P \lambda x. P(x) \land \text{FondOf}(x, y)$
9. Joe $\sim$ joe

10. Texas $\sim$ texas

11. Kaline $\sim$ kaline

12. Lockhart $\sim$ lockhart

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

(a)

```
S
  /\   \
DP |  VP
   /\   \
Joe V PP
      /\   \
   is P DP
      /\   \\  in  Texas
```

(b)

```
S
  /\   \
DP |  VP
   /\   \
Joe V AP
      /\   \
   is A PP
      /\   \\  fond of DP
           /\   \\  Kaline
```

(c)

```
S
  /\   \
DP |  VP
   /\   \
Kaline V DP
      /\   \
   is D N
      /\   \\
   a   cat
```
Exercise 5. Anni-Frid is a former millionaire does not entail Anni-Frid is a millionaire and *Anni-Frid is former. In this sense, former is a non-intersective modifier. Which of the following are non-intersective modifiers? Give examples to support your point.

(a) yellow
(b) alleged
(c) future
(d) intelligent
(e) good
(f) mere
6.2 Relative clauses

We turn now to another construction that uses the rule of Predicate Modification, namely relative clauses:

(7) woman who Björn loves

This expression characterizes an individual with two properties: (i) she is a woman; (ii) Björn loves her. In other words, this expression denotes (the characteristic function of) the intersection between the set of women and the set of individuals that Björn loves. This suggests a translation for the relative clause *who Björn loves* as:

\[ \lambda x. \text{Loves}(bj, x) \]

This predicate will be combined with \( \lambda x. \text{Woman}(x) \) via Predicate Modification to produce the right meaning, like so:

\[
\lambda x. [\text{Woman}(x) \land \text{Loves}(bj, x)]
\]

The question now becomes how we can compositionally derive translations like this for relative clauses.

As we have seen, the verb *loves* is transitive, so in ordinary, so-called ‘canonical’ sentences of English, this verb is followed by an object. But in this case, the relative pronoun *who*, which intuitively corresponds to the object of the verb, appears at the left edge of the relative clause *who Björn loves*. One way of understanding the connection between *who* and the object of *loves* is by assuming that there are (at least) two levels of syntactic representation, one where *who* occupies the canonical object posi-
tion immediately following the verb (‘Deep Structure’), and another where it has moved to its so-called ‘surface position’ (‘Surface Structure’). Under this view, wh- words like who (along with which, where, what, etc.) do not disappear entirely from their original positions; they leave a trace signifying that they once were there. The syntactic structure of the relative clause after movement would then be:

\[
\begin{aligned}
\text{CP} & \quad \langle e, t \rangle \\
\text{who}_i & \quad \text{C}' \\
\text{C} & \quad \text{S} \\
\text{DP} & \quad \text{VP} \\
\text{Björn} & \quad \text{V} \\
\langle e, \langle e, t \rangle \rangle & \quad \text{DP} \\
\langle e, \langle e, t \rangle \rangle & \quad e \\
\langle e, \langle e, t \rangle \rangle & \quad t_i \\
\end{aligned}
\]

where \( i \) is an arbitrary natural number, such as 1, 3, or 47. The subscript \( i \) on \textit{who} 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, and because the wh-word and the trace bear the same index, we say that the two expressions are co-indexed. 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′).\footnote{2}

In this structure, the C position is thought to be occupied by a silent version of the complementizer that. We hear the complementizer that instead of the relative pronoun who in, for example, woman that Björn loves. 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':

\begin{equation}
\text{good old-fashioned values} \left[\text{CP on which we used to rely}\right]
\end{equation}

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. Moreover, in some languages, including Bavarian German, relative pronouns can actually co-occur with complementizers \footnote{See Carnie (2013, Ch. 12) for a more thorough introduction to the syntax of relative clauses.}

\begin{equation}
\begin{align*}
\text{I woass ned } & \underline{\text{wann dass da}} \ \underline{\text{Xavea kummt.}} \\
\text{I know not } & \underline{\text{when that}} \ \underline{\text{the Xavea comes}} \\
\text{‘I don’t know when Xaver is coming’}
\end{align*}
\end{equation}

This provides additional evidence for the idea that relative pronouns like who and complementizers like that occupy distinct positions in relative clauses. To explain the fact that that and which cannot co-occur in English, we assume that either the relative pronoun or the complementizer that is deleted, in accordance with the ‘Doubly-Filled Comp Filter’ \footnote{Chomsky & Lasnik (1977)}, the principle that either the relative pronoun or that must be silent in English.\footnote{3}

The same kind of movement is thought to be true in a relative
clause like *who likes Agneta*, as in *singer who likes Agneta*, in which it is the subject, rather than the object, that is extracted, will have the trace in subject position thus:

\[
(10) \quad \text{CP} \\
\quad \text{who}_i \quad \text{C}' \\
\quad \text{C} \quad \text{S} \\
\quad \text{that} \quad \text{DP} \quad \text{VP} \\
\quad t_i \quad \text{V} \quad \text{DP} \\
\quad \text{likes} \quad \text{Agneta}
\]

In this tree, the relative pronoun *who* is co-indexed with a trace in the subject position for the embedded verb *likes*.

These syntactic assumptions lay the groundwork for a semantic treatment of relative clauses on which they function much like adjectival modifiers. The key assumptions are the following:

- Relative clauses are formed through a movement operation that leaves a trace.
- Traces 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 \( v_{i,\tau} \)
A trace may in principle correspond to a variable of any type. But in the cases we are considering at the moment, the type that fits best into the compositional machinery is type $e$. So we will assume by default that traces are type $e$, unless otherwise specified. The index on the trace should always correspond to the index on the variable. 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.

So, for example, the trace $t_7$ would be interpreted as the variable $v_{7,e}$, which we will abbreviate as just $v_7$:

$$t_7 \sim v_7$$

The denotation of the variable $v_7$ will then depend on an assignment; recall that $\begin{array}{c} v_7 \end{array}^{M,g} = g(v_7)$.4

We have thus arrived at a new composition rule:

**Composition Rule 4. Pronouns and Traces Rule**
If $\alpha$ is an indexed trace or pronoun, $\alpha_i \sim v_i$

---

4Contrast Heim and Kratzer’s rule, given in a direct interpretation style, where an assignment function decorates the denotation brackets: $\begin{array}{c} \alpha_i \end{array}^g = g(i)$. 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 $\begin{array}{c} \text{loves him}_i \end{array}^g = \lambda x. x \text{loves } g(i)$. 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:

\[\text{he}_7 \rightsquigarrow \nu_7\]

We will see more on the pronoun side of this in §6.3.

With these assumptions, we derive the representation

\[\text{Loves}(bj, \nu_1)\]

for Björn loves \(t_1\):

\[
\begin{align*}
\text{S} & \quad t \\
\text{Loves}(bj, \nu_1) & \\
\text{DP} & \quad \text{VP} \\
e & \quad \langle e, t \rangle \\
\text{bj} & \quad \lambda x. \text{Loves}(x, \nu_1) \\
\text{Björn} & \quad \text{V} \quad \text{DP} \\
\langle e, \langle e, t \rangle \rangle & \quad e \\
\lambda y \lambda x. \text{Loves}(x, y) & \quad \nu_1 \\
\text{loves} & \quad t_1
\end{align*}
\]

The translation corresponding to this S node, \(\text{Loves}(bj, \nu_1)\), is of type \(t\). Suppose that the complementizer *that* is a trivial identity function of type \(\langle t, t \rangle\), so \(\text{that} \rightsquigarrow \lambda p. p\), where \(p\) is a variable of type \(t\). So *that Björn loves \(t_1* has the same translation, of type \(t\). How does the relative clause end up with a meaning of type \(\langle e, t \rangle\)? In particular, how do we reach our goal, according to which the relative clause ends up with a translation equivalent to \(\lambda x. \text{Loves}(bj, x)\)?

\(^5\)The idea of treating traces and pronouns as variables is rather controversial; see Jacobson (1999) and Jacobson (2000) for critique and alternatives.
We can achieve our goal by assigning the relative clause an interpretation in which a lambda operator binds the variable \( v_1 \), thus:

\[
\lambda v_1 . \text{Loves}(x, v_1)
\]

In principle, the trace might have any index, so we need to know which variable to let the lambda-operator bind. We can do this with the help of the index of the relative pronoun. The rule of Predicate Abstraction, 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 \( \beta \)
- \( \beta \to \beta' \)
- \( \beta' \) is an expression of type \( t \)

Then \( \gamma' = \lambda v_i . \beta' \)

This gives us the following analysis of the relative clause:
We have reached our goal! The relative clause *that Björn loves* denotes the property of being loved by Björn. This can combine via Predicate Modification with *woman*, giving the property of being a woman who Björn loves, as it should.

Notice that under this analysis, the word *who* is never assigned a meaning. Rather, its contribution to the meaning lies in the fact that it triggers the rule of Predicate Abstraction, which gives a meaning for a tree.

**Exercise 6.** (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.
Exercise 7. Traditional grammar distinguishes between restrictive and non-restrictive relative clauses. Non-restrictive relative clauses are normally set off by commas in English, and they can modify proper names and other individual-denoting expressions.

1. Susan, who I like, is coming to the party.

2. *Susan who I like is coming to the party.

3. That woman, who I like, is coming to the party.

4. The woman who I like is coming to the party.

We have given a treatment of restrictive relative clauses in terms of Predicate Modification. Would an analysis using Predicate Modification in the same way be appropriate for non-restrictive relative clauses? Why or why not?
Exercise 8. For each node in the following tree, give the type and a fully $\beta$-reduced translation to $L_\lambda$.

You'll need to make an assumption about the meaning of the definite article *the*. For the purposes of this exercise, please assume that it is translated as follows:

$$\text{the} \rightarrow \lambda P. \iota x. P(x)$$

where $P$ is a predicate (type $(e, t)$), and $\iota x. P(x)$ is an expression of type $e$ that denotes the unique satisfier of $P$ (assuming there is one). So the type of the translation for *the* is $\langle \langle e, t \rangle, e \rangle$. We will justify this analysis in greater detail in Chapter 8.
6.3 Pronouns

Recall that the Pronouns and Traces Rule tells us that if \( \alpha \) is a trace or a pronoun, \( \alpha_i \sim v_i \). Thus pronouns and traces are interpreted in the same manner: as variables. In this section, we will try and justify this assumption.

**Exercise 9.** Using the pronouns and traces rule, give translations at every node for the following tree (ignoring the contribution of gender to the meaning):

\[
\begin{aligned}
S & \quad \downarrow \\
DP & \quad VP \\
\quad Sh_{3} & \quad V \\
\quad loves & \quad DP \\
\quad & \quad her_{5}
\end{aligned}
\]

Can all pronouns be interpreted as variables? For example, what if someone were to point to Cruella De Vil, and say:

(12) She is suspicious.

then *she* refers to Cruella De Vil. But I could point to Ursula and say the same thing, in which case *she* would refer to Ursula. I don’t have to point, of course; if Ursula is on TV then she is sufficiently salient for the same utterance to pick her out. Alternatively, I could raise Ursula to salience by talking about her:

(13) Ursula is usually mean, but offered to help Ariel. She is suspicious.

In this case, the pronoun is used **anaphorically**, as it has a linguistic antecedent. In the previous cases, the pronoun is used **deictically**.
Both the deictic and the anaphoric uses can be accounted for under the following hypothesis:

**Hypothesis 1.** All pronouns refer to whichever individual is most salient at the moment when the pronoun is processed.

(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, “Who/what does the pronoun refer to?”

(14) No woman blamed herself.
(15) Neither man thought he was at fault.
(16) Every boy loves his mother.
(17) the book such\(_1\) that Mary reviewed it\(_1\)

So not all pronouns are referential. Note that it is sometimes said that *No woman* and *herself* are “coreferential” in (14) but this is strictly speaking a misuse of the term “coreferential”, because, as Heim and Kratzer point out, *coreference implies reference.*

**Exercise 10.** 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 (14)–(17) can be analyzed as bound variables. For example, (14) should be translated as:

\[\neg \exists x. [\text{woman}(x) \land \text{blamed}(x, x)]\]
Furthemore, there are certain cases where pronouns behave almost identically to traces. For instance, regarding U.S. Supreme Court Justice Ruth Bader Ginzburg, it was once remarked:

(18) This is an older woman who everyone listens when she speaks.

The alternative with an unpronounced trace (*...who everyone listens when speaks) would have been ungrammatical; inserting the pronoun she rescues the sentence. Pronouns in such configurations are called resumptive pronouns. The semantic contribution of a resumptive pronoun is exactly like the semantic contribution of a trace: as a variable that is bound by a lambda operator. Thus

\[ \text{who everyone listens when she speaks} \]

denotes the property of being an \( x \) such that everyone listens when \( x \) speaks.

Another example in which pronouns are interpreted very much like traces is with such-relatives, as in:

(19) the book such that Mary read it

These cases can be treated much like relative clauses, using Predicate Abstraction. The trigger for the abstraction in this case is such, which is coindexed with a pronoun rather than a trace. For example, in (17), there is coindexation between such and it. The analysis works as follows:\(^6\)

---

\(^6\)Let us assume that \( x \) is a distinct variable from \( v_1 \).
Exercise 11. Give the types and a fully $\beta$-reduced logical translation for every node of the following tree.
For the definite article, please assume the treatment described in Exercise 8.

In light of this evidence, let us consider the possibility that pronouns should always be treated as bound variables.\[\text{\text{Heim & Kratzer (1998) pp. 116-118) define a distinction between free and bound variables in natural language in their ‘direct interpretation’ framework, where the interpretation is relative to an assignment function that applies directly to pronouns and traces, rather than their interpretations. Because we are using ‘indirect interpretation’, we can let the notions of ‘free’ and ‘bound’ for natural language expressions be inherited from their standard conceptions in logic.}}\]
Hypothesis 2. All pronouns are translated as bound variables.

What this means is that whenever a pronoun occurs in a sentence, the sentence translates to a formula in which the variable corresponding to the pronoun is bound. Currently, 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 (51) we would have to QR Ursula to a position where it QRs She in the second sentence somehow. It is not completely crazy to imagine that proper names can undergo QR, but it is 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.

(21) 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.
There are two readings of Peter Venkman’s quip, a sympathetic reading and a reading on which he is as usual being a jerk. On the sloppy reading (the sympathetic reading), Peter blames himself. On the strict reading (the asshole reading), Peter blames Egon. The strict/sloppy ambiguity exemplified in (21) 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:

(22)

S
  /\      /
 DP  LP   S
  /   \    /
 I  I 1
 |
 DP  VP
  |  |   |
 t1 V DP
     |    |
    blame myself1

The strict reading can be derived from an antecedent without QR:

(23)

S
  /\      /
 DP  VP   
  /   |   |
 I  V  DP
    |    |
   blame myself1

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
Other composition rules

*myself* is interpreted as co-referential with *I*, so a full explanation of this contrast awaits an answer to these questions.

**Exercise 12.** 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:

(24)

```
S
   ├── DP
   │    └── VP
   │        ├── She₁
   │        │    ├── V
   │        │    │    └── A
   │        │    └── is
   │             └── nice
```

(25)

```
S
   ├── DP
   │    └── VP
   │        ├── John
   │        │    └── V
   │        │        └── hates
   │        └── DP
   │            └── 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 13.** 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 con-
text. As [Heim & Kratzer (1998)] put it:

Treating referring pronouns as free variables implies a new way of looking at the role of variable assign-
ments. Until now we have assumed that an LF whose truth-value varied from one assignment to the next could *ipso facto* not represent a felicitous, complete utterance. We will no longer make this assumption. Instead, let us think of assignments as representing the contribution of the utterance situation.

Still, it is not appropriate to say *She left!* if your interlocutor has no idea who *she* refers to. We can capture this idea using dynamic semantics, discussed in Chapter 9.
7 Quantifiers

7.1 Quantifiers: Type \(\langle (e, t), t \rangle\)

7.1.1 Quantifiers: not type \(e\)

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

(1) a. A singer loves Anni-Frid.
   b. Anni-Frid loves a singer.

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

We also lack an analysis for the following sentences.

(2) Somebody is lazy.
(3) Everybody is lazy.
(4) Nobody is lazy.
(5) \{Some, every, at least one, at most one, no\} linguist is lazy.
(6) \{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?
We could arrive at type \( t \) at the S node by treating these expressions as type \( e \), like \textit{Anni-Frid}, \textit{Agneta}, and \textit{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:

\begin{equation}
(7) \text{ Susan came yesterday morning.} \\
\end{equation}

Therefore, Susan came yesterday.

This is a valid inference if the subject, like \textit{Susan}, denotes an individual. Here is why. Everything that came yesterday morning came yesterday. If \( \alpha \) denotes an individual, then \( \alpha \text{ came yesterday morning} \) 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 (2)–(6) fail to validate subset-to-superset inferences. For example:

\begin{equation}
(8) \text{ At most one letter came yesterday morning.} \\
\end{equation}

Therefore, at most one letter came yesterday.

This inference is not valid, so \textit{at most one letter} must not denote an individual.
Exercise 1. Which of the expressions in (5) 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:

(9) 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:

(10) 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 2. Which of the expressions in (5) fail to validate the law of contradiction? Give examples.
Exercise 3. 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:

(11)  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:

(12)  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 $e$.

7.1.2 Solution: Quantifiers

Let us recap. We assume that a VP denotes a predicate (type \( \langle e, t \rangle \)) and that a sentence denotes a truth value (type \( t \)). We have a bunch of expressions that are not of type \( e \), and we want them to combine with the VP to produce something of type \( t \). What can we do? The solution 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:

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

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:

\[
(13) \quad \text{something} \sim \lambda P. \exists x. P(x)
\]

In contrast, the function denoted by nothing returns true if there is nothing satisfying the predicate in the relevant model:

\[
(14) \quad \text{nothing} \sim \lambda P. \neg \exists x. P(x)
\]

The function denoted by everything returns true if everything satisfies the predicate:

\[
(15) \quad \text{everything} \sim \lambda P. \forall x P(x)
\]

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:
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 singer to function in the same way as everyone. The input (e.g. singer) 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:

\[
\begin{align*}
\text{some} & \sim \lambda P \lambda Q. \exists x. [P(x) \land Q(x)] \\
\text{no} & \sim \lambda P \lambda Q. \neg \exists x. [P(x) \land Q(x)] \\
\text{every} & \sim \lambda P \lambda Q. \forall x. [P(x) \rightarrow Q(x)]
\end{align*}
\]

These will yield analyses like the following:
Exercise 5. Give an analysis of *A singer loves Anni-Frid* 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 is some individual that is both a singer and someone who loves Anni-Frid. You may have to introduce a new lexical entry for *a*.

Exercise 6. 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)  
\[
\begin{array}{c}
S \\
| \\
DP & VP \\
| \\
| \textit{Everybody} & V \\
| \\
| & \textit{snores} \\
\end{array}
\]

(b)  
\[
\begin{array}{c}
S \\
| \\
DP & VP \\
| \\
| \textit{Somebody} & V & DP \\
| \\
| & \textit{hugged} & \textit{Ariel} \\
\end{array}
\]

(c)  
\[
\begin{array}{c}
S \\
| \\
DP & VP \\
| \\
| \textit{Everyone} & V & AP \\
| \\
| & \textit{is} & A \\
| \\
| & \textit{afraid} & P \\
| \\
| & & \textit{of} \\
| \\
| & & \textit{Ursula} \\
\end{array}
\]
Exercise 7. 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 8. In the early 70’s, cases of VP coordination as in Sam smokes or drinks were analyzed using CONJUNCTION REDUCTION,
Exercise 9. In the Algonquian language Passamaquoddy (spoken in Maine, United States, and New Brunswick, Canada), voice marking on the verb can affect which scope readings are available.
Quantifiers

for quantifiers (Bruening, 2001, 2008). For example, (19) and (20) differ in voice-marking and are true in different circumstances. (The morphological glosses have been simplified.)

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

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

In (19), the verb is in direct voice, and the agent of the verb hold corresponds to the bare noun skitap ‘man’, interpreted as an indefinite (‘a man’). The patient (the thing being held) corresponds to puhtaya ‘bottle’, which is associated with the universal quantifier psite ‘all’. Speakers of Passamaquoddy judge this sentence to be true in the situation on the right in Figure 7.1, but not in the situation on the left. (Images created by Benjamin Bruening for the Scope Fieldwork Project; see http://udel.edu/~bruening/scopeproject/materials.html)

In (20) 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 7.1 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?
7.2 Generalized Quantifiers

Treating quantifiers as type $\langle\langle e, t\rangle,\langle e, t, t\rangle\rangle$, 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.\(^1\)

7.3 Quantifiers in object position

7.3.1 Quantifier Raising

Clearly, *Everybody loves Björn* should be translated as:

$$\forall x. \text{Loves}(x, \text{bj})$$

and *Björn loves everybody* should be translated as

$$\forall x. \text{Loves}(\text{bj}, x)$$

The first case, with the quantifier in subject position, can be derived compositionally using the tools that we have:

---

\(^1\)Recent 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.
But the case with the quantifier in object position (Björn loves everybody) cannot be. Observe what happens when we try:

\[ \forall x. \text{Loves}(x, bj) \]

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

\[ \lambda x. \text{Loves}(x, bj) \]

\[ \lambda y \lambda x. \text{Loves}(x, y) \]

\[ \text{loves} \]

\[ \text{Björn} \]

\[ \lambda y. \lambda x. \text{Loves}(x, y) \]

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

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

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

\[ \text{loves} \]

\[ \text{everyone} \]

The transitive verb is expecting an individual, so the quantifier phrase cannot be fed as an argument to the verb. And the quantifier phrase is expecting an \( \langle e, t \rangle \)-type predicate, so the verb cannot be fed as an argument to the quantifier phrase. It is rather an embarrassment that this does not work. It is clear what this sentence means!

According to the assumptions we made so far, \textit{everybody} translates as:

\[ (21) \quad \lambda P. \forall x. P(x) \]

The appropriate value for \( P \) here would be a function that holds of an individual if Björn loves loves that individual:
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:

\[(23) \quad [\lambda P \forall x. P(x)](\lambda x. \text{Loves}(bj, x)) \equiv \forall x. \text{Loves}(bj, x)\]

Exercise 10. Simplify the following expression step-by-step:

\[
[\lambda Q. \forall x[\text{Linguist}(x) \rightarrow Q(x)]](\lambda v_1. \text{Offended}(j, v_1))
\]

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 \(\langle (e, t), t \rangle\)) 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 \((24a)\) is converted to the Logical Form representation \((24b)\)
(24) a. S
   DP       VP
   Björn  V     DP
   loves  everybody

b. S
   DP       LP
   everybody  1       S
   DP       VP
   Björn  V     DP
   loves  t₁

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.
Quantifiers

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 (af-
Transformations map from DS to SS, and from SS to PF and LF:

\[
\text{DS} \quad \text{SS} \quad \text{LF} \quad \text{PF}
\]

This is the so-called ‘T-model’, or (inverted) ‘Y-model’ of Government and Binding theory, motivated originally by \textit{Wasow (1972)} and \textit{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 \textit{covert}.

Many other transformational generative theories of grammar have been proposed over the years, of course (see \textit{Lasnik & Lohn-dal (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 \textit{(Pollard & Sag, 1994)} and Lexical-Functional Grammar \textit{(Bresnan, 2001)}; other approaches to quantifier scope are taken in conjunction with those syntactic theories.

\begin{exercise}
Derive a translation into lambda calculus for \textit{Beth speaks a European language}. Start by drawing the LF, assuming that \textit{a European language} undergoes QR. Assume also that
\end{exercise}

\footnote{Note that ‘Logical Form’ refers here to a \textit{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.}
the indefinite article *a* can denote what *some* denotes, that *European* and *language* combine via Predicate Modification, and that *speaks* is a transitive verb of type \( \langle e, \langle e, t \rangle \rangle \). You can do this in the Lambda Calculator.

**Exercise 12.** *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 13.** Provide a fragment of English with which you can derive truth conditions for the following sentences:

1. Every conservative congressman smokes.
2. No congressman who smokes dislikes Susan.
4. Susan dislikes every congressman.
5. Some congressman from every state smokes.
6. Every congressman respects himself.

The fragment should include:

- a set of syntax rules
- lexical entries (translations of all of the words into \( L_\lambda \))
• 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.

7.3.2 A type-shifting approach

Quantifier Raising is only one possible solution to the problem of quantifiers in object position. Another approach is to interpret the quantifier phrase in situ, i.e., in the position where it is pronounced. In this case one can apply a type-shifting operation to change either the type of the quantifier phrase or the type of the verb. This latter approach, using flexible types for the expressions involved, adheres to the principle of “Direct Compositionality”, which rejects the idea that the syntax first builds syntactic structures which are then sent to the semantics for interpretation as a second step. 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’ (1993) system, a type $(e, (e, t))$ predicate can be converted into one that is expecting a quantifier for its first or second argument, or both.

**Exercise 14.** What is the problem of quantifiers in object position, and what are the main approaches to solving it? Explain in your own words.

Hendriks defines a general type-shifting schema called **ARGUMENT RAISING** (not because it involves “raising” of a quantifier phrase to another position in the tree – it doesn’t – but because it “raises” the type of one of the arguments of an expression to a more complex type). We will focus on one instantiation of this schema, called **OBJECT RAISING**, defined as follows. Here and in the following, we will use $x$ for variables associated with the subject, and $y$ for those associated with the object, wherever possible.

**Type-Shifting Rule 2. Object raising**  
(RAISE-O)  
If an English expression $\alpha$ is translated into a logical expression $\alpha'$ of type $(e, (a, t))$, for any type $a$, then $\alpha$ also has a
translation of type \(\langle\langle e, t\rangle, t\rangle, \langle a, t\rangle\rangle\) of the following form:

\[
\lambda Q(\langle e, t\rangle) \lambda x_a. Q(\lambda y. \alpha'(y)(x))
\]

(unless \(Q, y\) or \(z\) occurs in \(\alpha'\); in that case, use different variables).

Using this rule, a sentence like *Björn loves everybody* can be analyzed as follows, without quantifier raising:

(25)

\[
\begin{array}{c}
t \\
\forall y. \text{Loves}(s, y) \\
\hline
e \\
\hline
\langle e, t\rangle \\
\hline
\lambda x. \forall y. \text{Loves}(x, y) \\
\hline
\langle\langle e, t\rangle, \langle e, t\rangle\rangle \\
\hline
\lambda Q(\langle e, t\rangle) \lambda x. Q(\lambda y. \text{Loves}(x, y)) \\
\hline
\langle\langle e, t\rangle, t\rangle \\
\hline
\lambda P. \forall y. P(y) \\
\hline
\lambda y \lambda x. \text{Loves}(x, y) \\
\hline
\text{loves} \\
\end{array}
\]

**Exercise 15.** Reproduce the tree in (25) using the Lambda Calculator, doing the \(\beta\)-reductions along the way. Since the Lambda Calculator does not support type-shifting, just treat the type shift as if it is a silent sister to *loves*. To represent the type variable \(a\), write it as \(\alpha\).
In some situations, it can be useful to apply type-shifting to subject arguments. One such situation stems from scope ambiguities as they occur in sentences with two quantifiers such as *Somebody likes everybody*. Lifting the verb using the Object Raising rule and then combining it with its two arguments results in the surface scope reading, i.e. the reading in which the subject existential takes scope over the object universal. This is shown in the following tree, where subscripts indicate the types of the variables.

(26)

\[
\begin{align*}
&t \\
&\exists x \forall y. \text{Loves}(x, y) \\
&\langle\langle e, t\rangle, t\rangle \\
&\lambda P. \exists x. P(x) \\
&\langle e, t\rangle \\
&\lambda x. \forall y. \text{Loves}(x, y) \\
&\langle\langle\langle e, t\rangle, t\rangle, \langle e, t\rangle\rangle \\
&\lambda Q(\langle e, t\rangle) \lambda x. Q(\lambda y. \text{Loves}(x, y)) \\
&\langle\langle e, t\rangle, t\rangle \\
&\lambda y \lambda x. \text{Loves}(x, y) \\
&\text{loves}
\end{align*}
\]

But what about the inverse scope reading, in which the object universal takes scope over the subject existential? It turns out that in order to generate this reading we need to lift both arguments of the verb. To do so, we first need to raise the subject, with a rule we will call Subject Raising. We then lift the verb using the Subject Raising and then the Object Raising rule and combine the resulting doubly-lifted with its two arguments.

---

**Type-Shifting Rule 3. Subject raising** *(RAISE-S)*

If an English expression $\alpha$ is translated into a logical expres-
Quantifiers

If \( \alpha' \) is a term of type \( \langle a, \langle b, t \rangle, t \rangle \) for any type \( a \), then \( \alpha \) also has a translation of type \( \langle a, \langle \langle b, t \rangle, t \rangle, t \rangle \) of the following form:

\[
\lambda y \lambda Q(\langle b, t \rangle). Q(\lambda x. \alpha'(y)(x))
\]

(unless \( y, Q \) or \( x \) is free in \( \alpha' \); in that case, use different variables).

This rule is the mirror image of the Object Raising rule above, in the sense that this rule alters the way that a transitive verb combines with its subject argument, while the Object Raising rule alters the way it combines with its object argument.

We are now ready to generate the inverse scope reading of \textit{Somebody likes everybody}.

To do so, we apply Subject Raising to the verb, followed by Object Raising:

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

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

\[
\lambda Q(\langle b, t \rangle). \forall y. Q(\lambda x. \text{Loves}(x, y))
\]

\[
\lambda y \lambda x. \text{Loves}(x, y)
\]

\[
\lambda y \lambda Q(\langle b, t \rangle). Q'(\lambda y. Q(\lambda x. \text{Loves}(x, y)))
\]

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

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

\[
\lambda y \lambda Q(\langle b, t \rangle). Q(\lambda x. \text{Loves}(x, y))
\]

\[
\langle e, \langle b \rangle \rangle
\]

\[
\lambda y \lambda x. \text{Loves}(x, y)
\]

\[
\lambda P. \forall y. P(y)
\]

\[
\forall y \exists x. \text{Loves}(x, y)
\]
**Exercise 16.** What happens if we apply Object Raising to the verb, followed by Subject Raising? Draw a derivation at the same level of detail as the tree in (27). Can the resulting reading also be generated in a simpler way?

In fact, Subject Raising and Object Raising are both instances of a general type-shifting schema that Hendriks defines. The general schema is as follows: If an expression has a translation $a'$ of type $\langle d, (b, \langle c, t \rangle) \rangle$, where $d$ and $c$ are possibly null sequences of types, then that expression also has translations of the following form, where $x$ and $z$ stand for possibly null sequences of arguments of the same length as $d$ and $c$ respectively:

$$
\lambda x d \lambda Q((b,t),y) \lambda z c [Q(yb[a(x)y](z))]
$$

(unless $x$, $y$, $z$, or $Q$ occur in $a'$; in that case, just use different variables of the same type).

This schema works in the following way, for a verb $\alpha$ that expects at least one argument, the “targeted argument” as we will call it. In the following examples, this argument will be of type $e$, but more generally it could be of any type; this is why the schema uses $b$ instead of $e$. The sequences $x$ and $z$ represent whatever arguments the verb applies to before and after it combines with the targeted argument. Suppose now that a verb has combined with all of the arguments in $x$ and that its next argument is not of the expected type (say $e$) but rather it is a quantifier $Q$ of type $\langle (e, t), t \rangle$. In that situation, the verb cannot apply to $Q$; and if there are more arguments coming up, i.e. if $z$ is nonempty (for example, if $Q$ is in object position, $z$ will contain a slot for the subject), $Q$ cannot apply to the verb either. Hendriks’ schema adjusts the entry and type of the verb $\alpha$ by replacing $e$ with $\langle (e, t), t \rangle$ so that $\alpha$ can apply to $Q$. The adjusted entry provides $\alpha$ with all of the arguments in $x$, then with a fresh variable $y$, and finally with all
remaining arguments in $\overline{z}$ (such as the subject); and finally it abstracts over $y$ and uses the quantifier $Q$ to bind it. This makes sure that the adjusted entry behaves just as the original entry for $\alpha$ would do if the quantifier $Q$ was raised above $\alpha$ and all of its arguments, leaving a trace corresponding to the variable $y$.

To give a concrete example, the Object Raising rule above results from applying this schema with $\overline{x}$ and $\overline{a}$ as null (because the verb does not apply to any arguments before it combines with the object), $b$ as $a$ (corresponding to the type of the object – typically type $e$), $\overline{z}$ as $z$ (because after combining with the object, the verb still expects to apply to the subject), and $\overline{c}$ as $e$ (because the subject is of type $e$). To get the Subject Raising rule, we instantiate Hendriks’ schema above by setting $\overline{x}$ to $x$, $\overline{a}$ to $a$, $b$ to $e$, and $z$ and $\overline{c}$ to null.

7.3.3 Putative arguments for the movement analysis

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 ‘inverse linking’, also known as ‘binding out of DP’. Here is a classic example:

(28) 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:

(29)

\[
\begin{array}{c}
S \\
\quad DP \\
\quad \quad \text{every basket} \\
\quad \quad 1 \\
\quad S \\
\quad \quad DP \\
\quad \quad \quad D \\
\quad \quad \quad \quad one \\
\quad \quad \quad NP \\
\quad \quad \quad \quad apple \\
\quad \quad \quad PP \\
\quad \quad \quad \quad \quad P \quad t_1 \\
\quad \quad \quad \quad \quad \quad in \\
\quad \quad VP \\
\quad \quad \quad \quad is \ rotten
\end{array}
\]

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 17.** Derive the translation for (29) compositionally using the rules we have introduced so far. You may need to introduce new lexical entries.

**Exercise 18.** Explain why the correct reading of (28) cannot be derived via the Object Raising type-shifting rule. Start by applying the Object Raising type-shifting rule to the preposition *in.*
Argument #3: Antecedent-contained deletion   Another argument that has been made in favor of QR is based on examples like this:

(30) 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 War and Peace:

(31) I read War and Peace before you did.

so the underlying structure would be:

(32) I read War and Peace before you did read War and Peace.

But what happens if we try to fill in the antecedent VP in a case like (30)? The VP read every novel that John did contains the ellipsis site. We might get an infinite regress if we tried to fill it in: 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 (30) is known as ANTECEDENT-CONTAINED DELETION.

But there is a solution! If the quantifier phrase in (30) 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 [30] can be accommodated.

**Exercise 19.** Give an analysis of (33) with fully $\beta$-reduced $\lambda$-expressions at each node. Use of the Lambda Calculator is encouraged.

**Exercise 20.** 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:

(34)  
   a. Mary blamed herself.  
   b. Mary blamed Mary.

But this is not the case when a reflexive pronoun is anaphoric to a quantifier:

(35)  
   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:

(36)  
   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 6.3):

\[(38) \quad \lambda y.\text{Blamed}(y, v_1)\]

The translation corresponding to the tree in (37) then gives an appropriate translation:

\[(39) \quad \forall x[\text{Woman}(x) \rightarrow \text{Blamed}(x, x)]\]

**Exercise 21.** 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 (39) 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):

\[(40) \quad \text{herself} \sim \lambda R(e, \langle e, t \rangle) \lambda x. R(x, x)\]

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 22.** Use the treatment of herself in (40) to derive a translation for every woman blamed herself.

**Exercise 23.** Does the lexical entry for herself in (40) 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 (41a) but not in (41b) the latter has only the implausible reading implying the existence of an “international woman”.

\[(41) \quad \begin{align*}
\text{a. John knows a woman from every country.} \\
\text{b. #John knows a woman who is from every country.}
\end{align*}\]

Similarly, extraction of which country is possible from a prepositional phrase modifier but not a relative clause modifier:
(42)  
  a. Which country does John know a woman from?  
  b. *Which country does John know a woman who is from?

Example (42b) 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 (41b) 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 (43a) but not in (43b), where the VP is coordinated with another VP. In (43b) we have only the implausible reading that every man has the same mother.

(43)  
  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 (44a) but not (44b), where there is coordination.

(44)  
  a. I wonder whom Mary gave birth to.  
  b. *I wonder whom Mary gave birth to and will eventually die.

The badness of (44b) 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.

7.4 Quantifiers, negation and adverbs

As we have seen, both type-shifting and QR can account for scopal ambiguity. Thus, (45) receives inverse scope if we apply QR to the object everybody. The surface scope reading can be modeled if we subsequently apply QR to somebody as well.

(45) Somebody loves everybody. $\exists > \forall; \forall > \exists$

This raises the question whether other scope-taking elements also give rise to scopal ambiguities. It turns out that this is not usually the case. As Ladusaw (1988) notes, negation, modals, and sentence adverbs can only take surface scope with respect to each other. Take this example, from Ladusaw (1977):

(46) a. Shelly usually doesn't do her homework. Usually > Not
    b. Shelly doesn't usually do her homework. Not > Usually

Each of these examples is unambiguous. In the first, the adverb usually can take scope only over negation, not under it; in the second, it is the other way around. This suggests that negation and adverbs are scopally fixed.

If we combine these elements with quantificational subjects, the result is sometimes scopally ambiguous. The following exam-
Quantifiers

Examples are from McCloskey (1997):

(47)  
   a. At least one player always loses. 
       \textsc{One} > \textsc{Always}; \textsc{Always} > \textsc{One} 
   b. Most guests might be late. 
       \textsc{Most} > \textsc{Might}; \textsc{Might} > \textsc{Most} 
   c. Every player didn't score. 
       \textsc{Every} > \textsc{Not}; \textsc{Not} > \textsc{Every} 

Example (47a) can mean that there is an unfortunate player who loses every time (surface scope), or that it is always the case at a potentially different player loses (inverse scope). Similarly, (47c) can mean that no player scored (surface scope), or that not every player scored (inverse scope), though the second reading requires stress on \textit{didn't} (Beghelli & Stowell 1997); see also Horn 1989, §4.3 and Büring (1997).

The lack of scopal interaction between adverbs (including negation) is typically assumed to indicate that operations such as QR and type shifting are not available for these adverbs. But then, the fact that scopal interaction between adverbs and quantificational noun phrases is available poses a dilemma: if there is no equivalent of QR available for these adverbs, it is not obvious how the subject quantifier in examples such as (47) is able to take scope under the adverb (McCloskey, 1997). QR has the effect of giving a noun phrase wider scope than it normally would have; for example, QR can allow a subject to take scope above an object that has itself undergone QR. But here the scope of the subject would have to be lowered, not raised.

To see this more clearly, consider this simple analysis of negation as taking scope between the subject and VP:

(48) \( \textit{doesn't} \sim \lambda P_{(e,t)} \lambda x. \neg P(x) \) 

A simple sentence like \textit{Lisa doesn't like broccoli} would then be analyzed as follows:
If the subject were *Everybody* instead of *Lisa*, then the only reading we would derive would be the surface scope reading, with the universal quantifier scoping over the negation. Raising the quantifier would not help to bring the quantifier under the scope of negation.

A possible way of deriving the inverse scope reading is to make use of the so-called **vP-INTERNAL SUBJECT HYPOTHESIS**, according to which the subject of a sentence originates in the specifier of a phrase that sits below negation, called a vP (‘little v P’), and moves to its surface position above negation before it is pronounced. Using this assumption, and a type \( \langle t, t \rangle \) meaning for negation (a ‘sentential’ treatment of negation), we can derive an inverse scope reading for the subject quantifier if it is interpreted in its original position. As Logical Form (LF) is the level of representation at
which quantifiers are thought to be interpreted, it is necessary to assume that the quantifier lowers down into its original position at LF, a process known as RECONSTRUCTION.

**Exercise 24.** With a tree, show a compositional derivation for the inverse scope reading of *Everybody doesn't like brocolli* using the vP-internal subject hypothesis and a sentential meaning for negation.
8  Presupposition

8.1 Introduction

There are no dubstep albums by Gottlob Frege (the logician who lived in the 1800s); he just did not make any. So the following sentence is not true:

(1) There are dubstep albums by Frege.

Its negation, naturally, is true:

(2) There are no dubstep albums by Frege.

This is how things usually are; if a sentence is not true, then its negation is true. But this is not always the case.

The following sentence, in which every combines with dubstep albums by Frege, is not felt to be true:

(3) Every dubstep album by Frege is famous.

Yet few would assent to its negation:

(4) Not every dubstep album by Frege is famous.

Thus neither the original sentence nor its negation is felt to be true. How can this be?

The answer is that every presupposes the existence of something satisfying the description it combines with. This presupposition is inherited by the negation. As Chierchia & McConnell-
Ginet (2000) write, “If A presupposes B, then A not only implies B but also implies that the truth of B is somehow taken for granted, treated as uncontroversial.” Furthermore,

If A presupposes B, then to assert A, deny A, wonder whether A, or suppose A – to express any of these attitudes toward A is generally to imply B, to suggest that B is true and, moreover, uncontroversially so. That is, considering A from almost any standpoint seems already to assume or presuppose the truth of B; B is part of the background against which we (typically) consider A.

Thus, if A presupposes B, then A, the negation of A, a yes/no question targeting A, and a conditional sentence in which A figures as the antecedent will all presuppose B as well. Observe that the following sentences also imply that Frege made at least one dubstep album:

(5) Is every dubstep album by Frege famous?
(6) If every dubstep album by Frege is famous, then I must be out of the loop.

Every member of this family of sentences shares the implication; this is characteristic of presupposition.

A word or construction that signals a presupposition is called a presupposition trigger. Other presupposition triggers include the quantifiers both and neither, factive adjectives like glad, factive verbs like know, possessives, exclusives like only, and the definite article. Besides every, here are some examples (where >> signifies ‘presupposes’):

(7) a. Neither candidate is qualified
   >> There are exactly two candidates
b. Ed is glad we won
   >> We won
c. Ed knows we won  
   >> We won  

d. Ed’s son is bald  
   >> Ed has a son  

e. Only Ed came  
   >> Ed came  

f. The balcony is lovely  
   >> There is a balcony

The definite article is the presupposition trigger that the theory of presupposition grew up around, so we will spend the next section reviewing that history, using the definite article as a focal point.

8.2 The definite article

So far, we have seen two types for determiners:

- \(\langle e, t \rangle, \langle e, t \rangle\) for the indefinite article \(a\) in predicative descriptions such as \(a\) singer in Agneta is a singer

- \(\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle\) for quantifiers

This section motivates a treatment of definite determiners with yet a third type, namely \(\langle \langle e, t \rangle, e \rangle\). In a phrase like \(the\) moon, 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.

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 descrip-
tions convey existence (that there is a princess, in this case), and uniqueness (that there is only one).

In “On Denoting”, Russell [1905] proposes to analyze definite descriptions on a par with the quantifiers we have just analyzed. He proposes that The princess smokes means ‘There is exactly one princess and she smokes’:

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

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

**Exercise 2.**

(a) Give a Russellian lexical entry for the. What is the type of the under your treatment?

(b) Show this lexical entry in action in the following tree:

```
S
   / \   /
DP VP
  /   /
D NP V
 /   /   /
the princess smokes
```
Strawson (1950), in a response to Russell entitled “On Referring”, agrees that definite descriptions signal existence and uniqueness of something satisfying the description, but he disagrees with Russell’s proposal that these implications are entailments. His argument centers around so-called **empty descriptions**: definite descriptions in which nothing satisfies the descriptive content. For example, since France is not a monarchy, *the king of France* is an empty description. Strawson writes,

> To say, “The king of France is wise” is, in some sense of “imply”, to *imply* that there is a king of France. But this is a very special and odd sense of “imply”. “Implies” in this sense is certainly not equivalent to “entails” (or “logically implies”).

Putting it another way:

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 you whether you thought that what he had just said was true, or was false; whether you agreed or disagreed

---

1 With “disguised assertion”, Strawson is alluding to Russell’s idea that the form of a sentence containing a definite description, where the definite description appears as a term, is misleading, and that the quantificational nature of definite descriptions is disguised by this form.
with what he had just said. I think you would be inclined, with some hesitation, to say that you did not do either; that the question of whether his statement was true or false simply did not arise, because there was no such person as the King of France. You might, if he were obviously serious (had a dazed, astray-in-the-centuries look), say something like: I'm afraid you must be under a misapprehension. France is not a monarchy. There is no King of France.

Strawson’s observation is that we feel squeamish when asked to judge whether a sentence of the form The F is G is true or false, when there is no F. We do not feel that the sentence is false; we feel that the question of its truth does not arise, as Strawson put it.

Why doesn’t the question of its truth arise? Because the sentence presupposes something that is false, namely that there is one and only one King of France. Only when the presuppositions of a sentence are met can it make enough sense to be true or false. In fact, one way of defining presupposition is just in this way:

(8) Semantic definition of presupposition
A presupposes B if and only if:
Whenever A is true or false, B is true.

Exercise 3. Recall the definition of entailment:

A entails B if and only if:
Whenever A is true, B is true.

Notice how similar this definition is to the semantic definition of presupposition. Consider the relationship between these two definitions. Is semantic presupposition a species of entailment? Or is it the other way around? Or neither? Explain your reasoning.
One way of implementing the idea that sentences might be neither true nor false is by introducing a third truth value. Under this strategy, along with ‘true’ and ‘false’, we have ‘undefined’ or ‘nonsense’ as a truth value. It turns out that having this third truth value makes the formal system a bit easier to set up, so we will adopt that strategy here. Let us use $\bot$ 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 $\bot$. Then the question becomes how we can set up our semantic system so that this is the truth value that gets assigned to a sentence with a false presupposition.

Intuitively, the reason that 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:
Now, what does Frege mean by “permissible”? One way of formalizing this idea is that the denotes a function of type $\langle \langle e, t \rangle, e \rangle$ that is only defined for input predicates that characterize one single entity. This function applies to a predicate, and if there is exactly one satisfier of that predicate, then the return value is that satisfier. But if there are zero satisfiers or multiple satisfiers, then the function simply does not return a value.

Another way of capturing the same intuition is to introduce a special ‘undefined individual’ of type $e$. We will adopt this approach here, using the symbol $\equiv e$ to denote this individual in our meta-language. Note that we are not adding this symbol to our logical representation language $L_\lambda$; rather we use $\equiv e$ in our meta-language to refer to this ‘undefined entity’ we are imagining, specifying this as the denotation for empty descriptions.\footnote{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’.} A definite
description of the form \textit{the }F\textit{will denote }\#_e \textit{if the number of satisfiers of }F\textit{is not exactly one.}

To formalize this idea, we must 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 producing \( \iota \)-expressions:

\begin{quote}
**Syntax rule: Iota**
If \( \phi \) is an expression of type \( t \), and \( u \) is a variable of type \( e \), then \( \iota u . \phi \) is an expression of type \( e \).
\end{quote}

The semantics of iota-expressions is defined as follows:

\begin{quote}
**Semantic rule: Iota**
\[ [\iota u . \phi]_{M,g} = \begin{cases} 
  d & \text{if } \{ k : [\phi]_{M,g[u \mapsto k]} = 1 \} = \{ d \} \\
  \#_e & \text{otherwise}
\end{cases} \]
\end{quote}

**Exercise 4.** 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:
(9) \( \text{the} \to \lambda P. \, \text{i} \, x \, P(\, x) \)

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.

(10) \[
\text{DP} \\
\text{\textit{e}} \\
\text{i} \, x \, \text{President}(\, x) \\
\langle (\text{e}, \text{t}), \text{e} \rangle \\
\lambda P. \, \text{i} \, x \, P(\, x) \\
\text{\textit{the}} \\
\text{D} \\
\text{NP} \\
\langle (\text{e}, \text{t}) \rangle \\
\text{president} \\
\langle (\text{e}, \text{t}) \rangle \\
\text{N} \\
\text{president} \\
\text{president}
\]

(11) \[
\llbracket \text{i} \, x \, \text{President}(\, x) \rrbracket_{M, g} \\
= \begin{cases} 
\text{d} & \text{if } \{ k : \llbracket \text{President} \rrbracket_{M, g}[x \to k] = 1 \} = \{ \text{d} \} \\
\not\equiv_{\text{e}} & \text{otherwise}
\end{cases}
\]

For example, in a model corresponding to the White House in 2014, where the only president in the relevant domain is Barack Obama, the denotation of the phrase would be Barack Obama.

Now, let us assume that a predicate like Bald only yields true or false for actual individuals, and yields the undefined truth value \( \not\equiv \) when given the undefined individual \( \not\equiv_{\text{e}} \) as input. So:

\[
\llbracket \text{Bald}(\, \alpha) \rrbracket_{M, g}
\]

will be \( \not\equiv \) if \( \llbracket \alpha \rrbracket_{M, g} = \not\equiv_{\text{e}} \), and 1 or 0 otherwise, depending on whether \( \llbracket \alpha \rrbracket_{M, g} \) is in the extension of Bald with respect to \( M \) and \( g \). Since (the translation of) the king of France (used today) would have \( \not\equiv_{\text{e}} \) as its denotation, (the translation of) The King of France
is bald would then have \( \neq \) as its denotation.

**Exercise 5.** Explain how this Fregean treatment of the definite article vindicates Strawson's intuitions.

**Exercise 6.** Using the assumptions above, compute a derivation for the following tree:

Chain of symbols

This exercise can be solved using the Lambda Calculator.

**Exercise 7.** Compute a derivation for the following tree according to Frege's intuitions, translating *square root* as a constant of type \( \langle e, \langle e, t \rangle \rangle \), and *four* as a constant of type \( e \):
This exercise can be solved using the Lambda Calculator.

**Exercise 8.** Assuming the following lexical entries:

1. \( \text{book} \rightarrow \lambda x. \text{Book}(x) \)
2. \( \text{on} \rightarrow \lambda x \lambda y. \text{On}(y, x) \)
3. \( \text{pillow} \rightarrow \lambda x. \text{Pillow}(x) \)

Which of the following trees gives the right kind of interpretation for *the book on the pillow*?

Explain your answer. You will find it helpful to annotate the nodes with their types (if not their fully beta-reduced translations as well), and consider the type at the top of the tree.
Let us consider another example. Beethoven wrote one opera, namely *Fidelio*, but Mozart wrote quite a number of operas. So in a model reflecting this fact of reality, the phrase *the opera by Beethoven* has a defined value. But *the opera by Mozart* does not. Consider what happens when *the opera by Mozart* is embedded in a sentence like the following:

\[(12) \text{ The opera by Mozart is beautiful.} \]

This might have the following translation:

\[
\text{Beautiful}(\exists x. [\text{Opera}(x) \land \text{By}(x, \text{mozart})])
\]

This sentence will denote $\neq$ in a model where there are multiple operas by Mozart, assuming that Beautiful yields the value $\neq$ when applied to an expression whose semantic value is $\neq_e$. So again, the undefinedness of the definite description “percolates up”, as it were, to the sentence level.

**Exercise 9.** Both *The king of France is bald* and *The opera by Mozart is boring* have an undefined value relative to the actual world, but for different reasons. Explain the difference.

---

\(^3\)Here 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 (Cop-\[pock, 2018\]). A number of other options are available as well, which go under the headings of ‘relativism’, ‘contextualism’, ‘absolutism’ and ‘expressivism’.
8.3 Definedness conditions

In the previous section, we saw one example of a presuppositional expression, namely the definite determiner *the*. We translated the definite determiner using $\iota$-expressions, which fail to denote when nothing satisfies the description. In that case, we assumed that the denotation was a special ‘undefined individual’, denoted $\#_x$.

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:

(13)  
\begin{enumerate}
\item Both candidates are qualified.
\item Neither candidate is qualified.
\end{enumerate}

If there are two candidates and both are qualified, then (13a) is clearly true and (13b) is clearly false. But if there is any number of candidates other than two, then it is uncomfortable to say that these sentences are true or false. This suggests that *both candidates* and *neither candidate* come with a presupposition that there are exactly two candidates.

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 *no* 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:

$$\partial[|P| = 2]$$

\(^4\) $|P| = 2$ is short for $\exists x \exists y [x \neq y \land P(x) \land P(y) \land \neg \exists z [z \neq x \land z \neq y \land P(z)]]$. 

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:

(14)  $\textit{neither} \sim \lambda P \lambda Q . \left[ \partial(\left| P \right| = 2) \land \neg \exists x . [P(x) \land Q(x)] \right]$

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 a 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:

**Syntax Rule: 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:

**Semantic Rule: Definedness conditions**

If $\phi$ is an expression of type $t$, then:

$$\left[ \partial(\phi) \right]^{M,g} = \begin{cases} 1 & \text{if } \left[ \phi \right]^{M,g} = 1 \\ \# & \text{otherwise.} \end{cases}$$

The lexical entry in (14) will give us the following analysis for (13b), where $\beta$-reduced variants of the translations are given at each node:
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 $\forall x. \ [\text{Candidate}(x) \rightarrow \text{Qualified}(x)]$.\(^5\)

The existence presupposition of the quantifier every can be treated using definedness conditions as well. We can capture it using the following kind of analysis of every:

(16) \hspace{1cm} \text{every} \sim \lambda P \lambda Q. [\partial(\exists x. P(x)) \land \forall x. [P(x) \rightarrow Q(x)]]

This will give rise to an undefined value for Every dubstep album by Frege in models where there are no dubstep albums by Frege (such as the one corresponding to reality), capturing the intuition that the sentence is neither true nor false.

**Exercise 10.** Notice that Anna isn’t the only genius does not presuppose that there is only one genius (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

\(^5\)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.
Table 8.1: Truth tables for the connectives in $\partial L$

<table>
<thead>
<tr>
<th>&amp;</th>
<th>T</th>
<th>F</th>
<th>#</th>
<th>∨</th>
<th>T</th>
<th>F</th>
<th>#</th>
<th>¬</th>
<th>T</th>
<th>F</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>#</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>#</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>#</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>#</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
</tbody>
</table>

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):

- $\text{the} \leadsto \lambda P \lambda x. \partial(|P| \leq 1) \land P(x)$
- $\text{only} \leadsto \lambda P \lambda x. P(x) \land \forall y[y \neq x \rightarrow \neg P(y)]$

Draw a tree for Anna isn’t the only genius 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 geniuses. (Note that the only genius 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.)

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 con-
joined (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:

(17) Every boy loves his cat.

This would be translated:

(18) $\forall x. [\text{Boy}(x) \rightarrow \text{Loves}(x, \iota y[\text{Cat}(y) \land \text{Has}(x, y)])]$ 

This formula will be true in a model where every element of $D$
satisfies the formula:

$$[\text{Boy}(x) \rightarrow \text{Loves}(x, \iota y[\text{Cat}(y) \land \text{Has}(x, y)])]$$

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 (1995a)] assign the value #
to a universal claim consisting of 1s and #s. [Haug (2013)] allows a
universal claim to be true as long as it contains no 0s, unless it is
*always* undefined.

As [Muskens (1995a)] discusses, however we set things up, it
should be the case that our universal quantifiers ‘match’ our treat-
ment 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:
Presupposition

\[
\left[ \forall x . \phi \right]_{M,g} = \begin{cases} 
1 & \text{if } \left[ \phi \right]_{M,g[x \mapsto k]} = 1 \text{ for all } k \in D \\
\# & \text{if } \left[ \phi \right]_{M,g[x \mapsto k]} = \# \text{ for some } k \in D \\
0 & \text{otherwise}
\end{cases}
\]

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 quantifier:

\[
\left[ \exists x . \phi \right]_{M,g} = \begin{cases} 
0 & \text{if } \left[ \phi \right]_{M,g[x \mapsto k]} = 0 \text{ for all } k \in D \\
\# & \text{if } \left[ \phi \right]_{M,g[x \mapsto k]} = \# \text{ for some } k \in D \\
1 & \text{otherwise}
\end{cases}
\]

So an existential claim is true only if the scope proposition never takes on an undefined value, and is not always false.

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, \textbf{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 \( \left[ \alpha \right]_{M,g} = \left[ \beta \right]_{M,g} \), 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 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):

(19) The golden mountain does not exist.

One possible treatment of this sentence is with an existence predicate, which is true of actual individuals and false of undefined individuals. Let us name this predicate Exists. Then the translation of (19) would be:

(20) $\neg \text{Exists}(\iota x. [\text{Golden}(x) \land \text{Mountain}(x)])$

(This of course is not the only possible treatment of this example.) The semantics of Exists might then be defined as follows:

**Semantic rule: Existence predicate**

$[\text{Exists}(\alpha)]^{M,g} = 1$ if $[\alpha]^{M,g} \neq \#_e$ and 0 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 \( g(u) \in D_\tau \). We use \( g[x \rightarrow d] \) to denote an assignment function which is exactly like \( g \) with the possible exception that \( g(x) = d \).

The semantic rules are the following.

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

   (b) If \( \alpha \) is a variable, then \( [\alpha]^{M,g} = g(\alpha) \).

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

3. **Equality**
   
   If \( \alpha \) and \( \beta \) are terms, then
   
   \[
   [\alpha = \beta]^{M,g} = \begin{cases} 
   1 & \text{if } [\alpha]^{M,g} \neq # e \text{ and } [\beta]^{M,g} \neq # e \text{ and } [\alpha]^{M,g} = [\beta]^{M,g} \\
   # & \text{if } [\alpha]^{M,g} = # e \text{ and } [\beta]^{M,g} = # e \\
   0 & \text{otherwise} 
   \end{cases} 
   \]

4. **Connectives**
   
   The semantics of the connectives is defined as in Table 8.1.

5. **Quantification**
(a) If $\phi$ is a formula and $v$ is a variable of any type: then

$$\left[ \forall v \phi \right]^{M,g} = \begin{cases} 
1 & \text{if } \left[ \phi \right]^{M,g[v \mapsto k]} = 1 \text{ for all } k \in D \\
# & \text{if } \left[ \phi \right]^{M,g[v \mapsto k]} = # \text{ for some } k \in D \\
0 & \text{otherwise}
\end{cases}$$

(b) If $\phi$ is a formula and $v$ is a variable of any type: then

$$\left[ \exists v \phi \right]^{M,g} = \begin{cases} 
0 & \text{if } \left[ \phi \right]^{M,g[v \mapsto k]} = 0 \text{ for all } k \in D \\
# & \text{if } \left[ \phi \right]^{M,g[v \mapsto k]} = # \text{ for some } k \in D \\
1 & \text{otherwise}
\end{cases}$$

6. Lambda Abstraction

If $\alpha$ is an expression of type $\tau$ and $u$ a variable of type $\sigma$ then $\left[ \lambda u . \alpha \right]^{M,g}$ is that function $h$ from $D_\sigma$ into $D_\tau$ such that for all objects $k$ in $D_\sigma$, $h(k) = \left[ \alpha \right]^{M,g[u \mapsto k]}$.

This is just one example of a complete system; other design choices are of course possible.

8.4 Projection problem

The treatment of presupposition we have given so far captures the fact 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 Karttunen [1973] discussed. Consider the following examples:

(21) If there is a king of France, then the king of France is wise.
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:

If France is neutral, then the king of France is wise.

Either France is lucky or the king of France is wise.

Karttunen gave the following generalization:

When the antecedent of the conditional (the if-part) entails a presupposition of the consequent (the then-part), the presupposition gets filtered out.

In, 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, 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:

A presupposition of one disjunct gets filtered out when the negation of another disjunct entails it.

In, for example, the second disjunct (the king of France is wise) presupposes that there is a king of France. The first disjunct
is *there is no king of France*, whose negation is *there is a king of France*, which again entails of course that there is a king of France, so the presupposition gets filtered out. In (24) the first disjunct does not entail that there is a king of France, so the presupposition does not get filtered out.

Observe that generalizations (2) and (3) use the word *entail*. In (21) and (22) the part of the sentence that is supposed to entail the presupposition is simply equivalent to the presupposition. But it could also be stronger, and *strictly entail* the presupposition. Consider the following example from Karttunen (1973):

(27) Either Geraldine is not a catholic or she has stopped attending services on Sundays.

The second disjunct (*she has stopped attending services on Sundays*) presupposes that Geraldine did attend services on Sundays. The local context for the second disjunct is the negation of the first disjunct. The first disjunct is *Geraldine is not a catholic*, so the local context is *Geraldine is a catholic*. If we assume that all catholics attend services on Sundays, then the antecedent entails that Geraldine attends services on Sundays. So Karttunen's generalization correctly captures the fact that (27) does not presuppose that Geraldine attends services on Sundays.

The system that we have introduced for dealing with presuppositions might seem to predict that presuppositions will *always* project, since undefinedness tends to “percolate up,” so to speak. The projection problem has been dealt with elegantly using *dynamic 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 chapter.
9  |  Dynamic semantics

9.1  Introduction

In this chapter, we motivate dynamic semantics\[^1\] where the meaning of an utterance is something that depends on and updates the current discourse context. We will show first that the presupposition projection problem receives an insightful solution under a dynamic perspective on meaning. We then discuss pronouns with indefinite antecedents, including the famous ‘donkey sentences’:

\[(1)\quad \text{If a farmer owns a donkey, then he beats it.}\]

The chapter ends with a compositional dynamic fragment.

9.2  Presupposition in dynamic semantics

Recall the following generalizations from the previous chapter:

\[(2)\quad \text{When the antecedent of the conditional (the \textit{if}-part) entails a presupposition of the consequent (the \textit{then}-part), the presupposition gets filtered out.}\]

\[(3)\quad \text{A presupposition of one disjunct gets filtered out when the negation of another disjunct entails it.}\]

\[^1\text{Heim} 1982\text{b, 1983\text{b,a}; Kamp & Reyle 1993; Groenendijk & Stokhof 1990\text{a, 1991}; Muskens 1996 among others}\]
These two generalizations can be stated concisely and illuminatingly using Karttunen's (1974) concept of local context: In general, a presupposition gets filtered out if it is entailed by the appropriate local context. The local context for the consequent of a conditional is its antecedent, and the local context for one disjunct of a disjunction is the negation of the other disjunct.

This idea builds on Stalnaker's (1978) ideas about the pragmatics of presupposition. Stalnaker introduces the concept of the context set, which is conceived of as the set of 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 $\phi = \psi$) 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 worlds. But there are also other worlds where Lisa, who is also a child, is the president. Call these $w_4$ and
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 399) puts it:

A context is here construed more or less... as a set of propositions, or more simply, as a proposition, namely that proposition which is the conjunction of all the elements of the set.

If propositions are sets of possible worlds, then what is the conjunction of a set of propositions? Here is a concrete example:

\[
\begin{align*}
P &= \{ w_1, w_2, w_3 \} & [\text{‘Bart is president’}] \\
Q &= \{ w_1, w_2, w_3, w_4, w_5 \} & [\text{‘A child is president’}] \\
R &= \{ w_2, w_3 \} & [\text{‘Bart has a girlfriend’}] \\
S &= \{ w_1, w_4 \} & [\text{‘Lisa is sick’}] \\
W &= \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10} \} & [\text{the set of all worlds}]
\end{align*}
\]

What is the conjunction of \( P \) and \( S \), the conjunction of the proposition that Bart is president and the proposition that Lisa is sick? It is the set of worlds where both propositions are true. That is the intersection (not the union).

\[
\begin{align*}
P \cap S &= \{ w_1, w_2, w_3 \} \cap \{ w_1, w_4 \} \\
&= \{ w_1 \}
\end{align*}
\]
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 update crashes when the presuppositions of the sentence are not satisfied. Recall from Chapter 9 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 *John is happy*, for example, would eliminate all worlds where John is not happy from the context set.

**Exercise 1.** Suppose that the context \( c \) consists of the following worlds: \( \{ w_1, w_2, w_3, w_4, w_5 \} \) and in these worlds it is raining: \( \{ w_2, w_4 \} \). What is the result of updating \( c \) with *It is raining*?

Suppose that we have a sentence like *John's son is bald*, which presupposes that John has a son. If there are some worlds in the context set where John does not have a son, then the presuppositions of the sentence are not satisfied in the context set. In such a situation, we say that the context set does not admit the sentence. Karttunen’s idea is that in order for a context to admit a sentence, the context must entail the presuppositions of the sentence. Admittance is defined in terms of satisfaction:

\[(4) \text{ Satisfaction} \]

Let \( P_\phi \) be the set of worlds where the presuppositions of \( \phi \) are satisfied. A context \( c \) satisfies the presuppositions of \( \phi \) if \( X \subseteq P_\phi \).
(5) **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 presuppositions are taken for granted; this process is called **GLOBAL ACCOMMODATION**.\(^2\) But the idea is nevertheless that the update cannot proceed until the context is such that all the presuppositions of the sentence are satisfied.

A simple, non-compound sentence will have a set of *basic presuppositions*. For example, *Both Homer’s children are bald* presupposes that Homer has exactly two children; this is a basic presupposition of this non-compound sentence. Non-compound sentences are admitted by a context as long as the context entails all of their basic presuppositions:

(6) **Admittance conditions for non-compound sentences**

If $\phi$ 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 2.** 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']
- $S = \{ w_1, w_4 \}$ ['Lisa 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 $\phi = \text{Both Homer’s children are bald}$

---

\(^2\)As opposed to **LOCAL ACCOMMODATION**, which has been posited as a last-resort mechanism that allows interpreting a presupposition locally under semantic operators when it cannot project for some reason. We will not discuss local accommodation in this book.
presupposes that Homer has exactly two children. Suppose that in worlds $w_1...w_8$, only Bart and Maggie are children of Homer, but in $w_9$ and $w_{10}$, Bart, Lisa, and Maggie are. So the proposition that Homer has exactly two children (call it $K$) is the set of worlds $w_1...w_8$:

$$K = \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8 \}$$

$K$ is a basic presupposition of $\phi$; 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:

(7) If baldness is hereditary, then John’s son is bald.
    $\Rightarrow$ John has a son.

(8) If John has a son, then John’s son is bald.
    $\not\Rightarrow$ John has a son.

In the first example, the sentence as a whole presupposes that John has a son (as indicated by the symbol $\Rightarrow$). In the second example, the sentence as a whole does not at all convey that the speaker believes that John has a son. The speaker appears quite open to the possibility that he does not. Again, in a conditional sentence of the form *If $A$ then $B$*, if the antecedent $A$ satisfies the presuppositions of $B$, then the conditional as a whole does not carry the presuppositions of $B$. 
Karttunen (1974) makes sense of this by imagining that we first update the global discourse context with $A$, and that it is in this temporary, hypothetical context that the presuppositions of $B$ have to be satisfied. For conditionals, Karttunen proposes the following:

(9) **Admittance conditions for a conditional sentence**

Context $c$ admits “If $\phi$ 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:

(10) If Homer has exactly two children, then both his children are bald.
(11) If Homer is bald, then both his children are bald.

Assume the following:

\[
A = \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8 \} \quad \text{[‘Homer has exactly 2 children’]}
\]
\[
W = \{ w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10} \} \quad \text{[the universe]}
\]

Suppose that $c = W$. Does $c$ admit (10)? According to the admittance conditions for conditional sentences in (9), it does just in case (i) $c$ admits *Homer has exactly two children* and (ii) $c + \text{Homer has exactly two children}$ admits *both his children are bald*. Since *Homer has exactly two children* carries no presuppositions, the first condition is satisfied. What about the second condition? The result of updating $c$ with *Homer has exactly two children* is the set $A$, the set of worlds where Homer indeed has exactly two children. Now the question is whether this set, $A$, admits the non-
Dynamic semantics

compound sentence *both his children are bald*. Since it is a non-compound sentence, the rule for non-compound sentences \[6\] applies. What *both his children are bald* presupposes is that Homer has exactly two children. This is satisfied in all of the worlds in \(A\), so the second condition is satisfied as well. Hence \(c\) does admit \(\text{(10)}\). But the same does not hold for \(\text{(11)}\).

**Exercise 3.** Explain step-by-step why \(c\) (as defined in the foregoing discussion) does not admit \(\text{(11)}\).

**Exercise 4.** Refer to Figure \[9.1\]. Does \(C\) admit *If the king has a son, then the king’s son is bald*? Why or why not? Does \(K\) admit it? Why or not? Explain using the definition of admittance, and assume that the result of updating a context with *The king has a son* is the intersection of \(A\) with the context *if the presuppositions of ‘The king has a son’ are satisfied*; undefined otherwise.

So now we are in a position to explain why the presupposition that Homer has at least three children *projects* in a case like \(\text{(11)}\) and not in a case like \(\text{(10)}\). In order for the conditional as a whole to be admitted by a given context, both of the conditions in \(\text{(9)}\) 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 me 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 \(\text{(11)}\) where the antecedent of the conditional does *not* entail the presuppositions of the consequent. In order for that sentence to be admitted in a given context, the con-
text 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:

(12) A context $c$ admits a sentence $S$ just in case each of the constituent sentences of $S$ is admitted by the corresponding local context. ([Heim 1983e, 399])

For example, consider (27) above, repeated here:

(13) Either Geraldine is not a catholic or she has stopped attending services on Sundays.

Here we have a disjunction. The local context for the second disjunct is $c+$ the negation of the first disjunct. The first disjunct ($Geraldine$ is not a catholic) is itself negated; let us assume that the negation of the negated sentence can be obtained simply by removing the ‘not’, so the local context for the second disjunct is $c+$ Geraldine is a catholic. Suppose it is common ground in the global context that all catholics attend services on Sundays. Then the local context entails that Geraldine attends services on Sundays. The consequent, she has stopped attending services on Sundays, presupposes that Geraldine attends services on Sundays. Since the local context entails this proposition, the global context need not entail it, so the presupposition is filtered out.

**Exercise 5.** Give another example of a disjunction in which the negation of the antecedent entails the presuppositions of the consequent, and explain how the presuppositions of the consequent get filtered out.

### 9.3 Pronouns with indefinite antecedents

Another important motivation for dynamic semantics comes from pronouns with indefinite antecedent. In dynamic semantics, an indefinite noun phrase like a man introduces a new DISCOURSE
REFERENT into the context, and an anaphoric pronoun or definite description picks up on the discourse referent.

One of the main motivations for dynamic semantics comes from examples involving pronouns whose antecedents are indefinite descriptions, as in the following two-sentence discourse:

(14) My neighbor found a cat. Then it ran away.

So far, we have analyzed indefinite descriptions as existential quantifiers. This was Russell’s (1905) treatment.

There are good reasons to favor Russell’s treatment of indefinites over one on which indefinites refer to some individual, as Heim (1982b) discusses. First, it correctly captures the fact that (15) does not imply that there is a dog that John and Mary are both friends with a dog.

(15) John is friends with a dog and Mary is friends with a dog. If we assumed that a dog referred to some dog, then we would predict this sentence to have that implication. Second, Russell’s analysis correctly captures the fact that (16) does not say regarding some particular dog that it came in, in contrast to (17) which has a proper name referring to a dog and does have that implication.

(16) It is not the case that a dog came in.
(17) It is not the case that Fido came in.

Thirdly, Russell’s analysis correctly captures the fact that (18) can be true even if it is not the case that there is some particular dog that everybody owns, while (19) does not have that implication.

(18) Every child owns a dog.
(19) Every child owns Fido.

If a dog referred to a dog then (18) should mean that every child owns that dog, as in (19).

However, there are some problems. If we analyze example (14)
using Russell’s very sensible analysis, we will derive the following
representation (assuming that it carries the index 3, and that a se-
quence of two sentences is interpreted as the conjunction of the
two sentences):

\[ (20) \exists x [\text{Cat}(x) \land \text{Found}(j, x)] \land \text{RanAway}(v_3) \]

with \( v_3 \) an unbound variable outside the scope of the existential
quantifier. (It doesn't matter which variable we choose; even if
we choose \( x \), the variable will still be unbound, because it will be
outside the scope of the existential quantifier.) Assuming that QR
does not move quantifiers beyond the sentence level, the scope of
the existential quantifier introduced by \( a \) cat does not extend all
the way to include the variable \( v_3 \), and there is no other variable-
binder to bind it.

**Exercise 6.** Give LF trees and derivations for the two sentences in
(14). (Feel free to treat ran away as a single verb.) Explain why
these representations do not capture the connection between the
connection between the pronoun and its intuitive antecedent.

One imaginable solution to this problem is to allow QR to move
quantifiers to take scope over multiple-sentence discourses, so we
could get the following representation:

\[ (21) \exists x [\text{Cat}(x) \land \text{Found}(s, x) \land \text{RanAway}(x)] \]

Regarding this imaginable solution, Heim (1982a, 13) writes the
following:

This analysis was proposed by Geach [1962, 126ff]. It
implies as a general moral that the proper unit for the
semantic interpretation of natural language is not the
individual sentence, but the text. [The formula] pro-
vides the truth condition for the bisentential text as
a whole, but it fails to specify, and apparently even precludes specifying, a truth condition for the [first] sentence.’

Heim[1982a] also presents a number of empirical arguments against this kind of treatment. One comes from dialogues like the following:

(22) a. A man fell over the edge.
    b. He didn't fall; he jumped.

(23) a. A dog came in.
    b. What did it do next?

What would a Geachian analysis be for a case like (22)? If we let the existential quantifier take scope over the entire discourse, we would get the 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 (23) presents a similarly puzzling challenge.

Another argument that Heim makes against the Geachian analysis is based on the following example:

(24) Susan owns some sheep. Harry vaccinated them.

This sentence should be false in a situation where Susan owns six sheep, of which Harry vaccinated three. On the Geachian analysis, the interpretation would be something along the lines, ‘there exists an $x$ such that $x$ is a bunch of sheep and Susan owns $x$ and Harry vaccinated $x$’, which would be true in such a situation. But the English sentence would not be, so this is not a welcome prediction.

Thirdly, Geach’s proposal would mean that existential quantifiers have different scope properties from other quantifiers. Consider the following examples:

(25) A dog came in. It lay down under the table.
(26) Every dog came in. #It lay down under the table.

(27) No dog came in. #It lay down under the table.

In neither (26) nor (27) can it be bound by the quantifier in the first sentence.\footnote{Heim 1992, 17} concludes:

The generalization behind this fact is that an unembedded sentence is always a “scope-island,” i.e. a unit such that no quantifier inside it can take scope beyond it. This generalization (which is just a special case of the structural restrictions on quantifier-scope and pronoun-binding that have been studied in the linguistic literature) is only true as long as the putative cases of pronouns bound by existential quantifiers under Geach’s analysis are left out of consideration.

Thus it seems that Geach’s solution will not do, and we need another alternative.

So-called ‘donkey anaphora’ is another type of case involving pronouns with indefinite antecedents that motivates dynamic semantics. The classic ‘donkey sentence’ is:

(28) If a farmer owns a donkey, then he beats it.

This example is naturally interpreted as a universal statement, representable as follows:

(29) $\forall x \forall y [[\text{Farmer}(x) \land \text{Donkey}(y) \land \text{Own}(x, y)] \rightarrow \text{Beats}(x, y)]$

\footnote{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”)}
But the representation that we would derive for it using the assumptions that we have built up so far would be:

\[(30) \quad [\exists x \exists y [\text{Farmer}(x) \land \text{Donkey}(y) \land \text{Own}(x, y)] \rightarrow \text{Beats}(x', y')]\]

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:

(31) Every man who owns a donkey beats it.

**Exercise 7.** Give a representation in $L_\lambda$ capturing the intuitively correct truth conditions for (31). Then give an LF tree and a derivation for (31) using the assumptions that we have built up so far. Does this derivation give an equivalent result? If so, explain. If not, give a situation (including a particular assignment function) where one would be true but the other would be false.

According to Geach (1962), we must simply stipulate that indefinites are interpretable as universal quantifiers that can have extra-wide scope when they are in conditionals or in a relative clause. But this is more of a description of the facts than an explanation for what is happening. Moreover, it is not as if just any relative clause allows for a wide-scope universal reading of an indefinite within it:

(32) A friend of mine who owns a donkey beats it. There is no wide-scope universal reading for *a donkey* here.

Heim’s (1982b) idea is that indefinites have no quantificational force of their own, but are 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:
(33) In most cases, if a table has lasted for 50 years, it will last for 50 more. 
\[\iff \text{Most tables that have lasted for 50 years will last for another 50.}\]

(34) Sometimes, if a cat falls from the fifth floor, it survives. 
\[\iff \text{Some cats that fall from the fifth floor survive.}\]

(35) If a person falls from the fifth floor, he or she will very rarely survive. 
\[\iff \text{Very few people that fall from the fifth floor survive.}\]

However, on Heim’s view, indefinites are unlike pronouns in that they introduce a ‘new’ referent, while pronouns pick up an ‘old’ referent. This idea of novelty is formulated in the context of dynamic semantics, where as a sentence or text unfolds, we construct a representation of the text using discourse referents. A pronoun picks out an established discourse referent. An indefinite contributes a new referent, and has no quantificational force of its own. The quantificational force arises from the indefinite’s environment.

The idea of a **discourse referent** is laid out by [Karttunen (1976)](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 pro-
noun or a definite noun phrase later in the text.\footnote{4} Thus a discourse referent need not correspond to any actual individual; in this sense, a discourse referent does not necessarily imply a referent. There are examples in which the occurrence of a coreferential pronoun or definite noun phrase is justified, but no particular individual is talked about, as in *No man wants his reputation dragged through the mud.* A discourse referent is more like a placeholder for an individual, very much like a variable. According to Karttunen, one of the virtues of this notion is that it “allows the study of coreference to proceed independently of any general theory of extralinguistic reference” (p. 367).

Karttunen\footnote{1976} also pointed out that discourse referents have a certain *lifESPAN*; they do not license subsequent anaphora in perpetuity. Here is an example where a discourse referent dies:

(36) Susan didn’t find a cat and keep it. #It is black.

The pronoun *it* in the second sentence cannot refer back to the discourse referent that the *it* in the first sentence picks up. The lifespan of that discourse referent ends with the scope of negation. Examples \([26]\) and \([27]\) above provide further examples in which one can see evidence of lifetime limitations for discourse referents. So while indefinites seem to introduce discourse referents with an unusually long life span, compared to other apparently quantificational expressions, the discourse referents they introduce aren’t immortal. A good theory should account for both sides of this tension.

### 9.4 File change semantics

Heim’s\footnote{1982b} FILE CARD SEMANTICS conceptualizes discourse ref-
erents as file cards, very much building on Karttunen’s metaphor. In file card semantics, an indefinite introduces a new file card. Subsequent anaphoric reference updates the file card. For example, consider the discourse in (37):

(37)  

a. A dog bit a woman.  
b. She hit him with a paddle.  
c. It broke in half.  
d. The dog ran away.

The first sentence contains two indefinites, *a dog* and *a woman*. These trigger the introduction of two new file cards; call them file card 1 and file card 2. File card 1 is associated with the property ‘dog’, and ‘bit 2’, and file card 2 is associated with the property ‘woman’, and ‘bitten by 1’. Pictorially, we can represent the situation like this:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>woman</td>
</tr>
<tr>
<td>bit 2</td>
<td>bitten by 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>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>woman</td>
<td>paddle</td>
</tr>
<tr>
<td>bit 2</td>
<td>bitten by 1</td>
<td>used by 2 to hit 1</td>
</tr>
<tr>
<td>was hit by 2 with 3</td>
<td>hit 1 with 3</td>
<td></td>
</tr>
</tbody>
</table>

And so forth, so that by the end of the discourse, the file looks like this:

(38)  

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>woman</td>
<td>paddle</td>
</tr>
<tr>
<td>bit 2</td>
<td>bitten by 1</td>
<td>used by 2 to hit 1</td>
</tr>
<tr>
<td>was hit by 2 with 3</td>
<td>hit 1 with 3</td>
<td></td>
</tr>
<tr>
<td>ran away</td>
<td>broked in half</td>
<td></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 8.** Add a sentence to (37) and show what the file would look like afterwards.

Like Karttunen, Heim wishes to distinguish between discourse referents (i.e., file cards) and the things that they talk about. She reasons that such an identification would be absurd, because a file card is just a description and in principle it could match any number of individuals.

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 (38) depends on whether it is possible to find a sequence of individuals that match the descriptions on the cards. For example, consider the following two worlds. Assume that in both worlds, Joan and Sue are women, Fido and Pug are dogs, and Paddle is a paddle.

**World 1**
- Pug bit Joan
- Joan hit Pug with Paddle
- Paddle broke in half
- Pug ran away

**World 2**
- Fido bit Joan
- Joan hit Fido with Paddle
- Paddle broke in half
- Fido ran away
In both worlds, it is possible to find a sequence of individuals that match the descriptions. In World 1, the sequence is \{Pug, Joan, Paddle\} (corresponding respectively to file cards 1, 2, and 3), and in World 2, it is \{Fido, Joan, Paddle\}. So the file is true relative to both worlds.

More technically, we say that a given sequence of individuals satisfies a file in a given possible world if the first individual in the sequence fits the description on card number 1 in the file (according to what is true in the world), the second individual fits the description on card 2, etc. A file is true (a.k.a. satisfiable) in a possible world if and only if there is a sequence that satisfies it in that world.

On this view, the 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 \{Pug, Joan, Paddle\} would be in the set of world-sequence pairs making up the file represented by (38). So would the pair consisting of World 2 and the sequence \{Fido, Joan, Paddle\}. As the meaning of a sentence in a 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.

### 9.5 Discourse Representation Theory

File card semantics is not the only dynamic theory of meaning; another very well-developed and well-known one is **DISCOURSE REPRESENTATION THEORY** ([Kamp & Reyle, 1993](#)), in which **DISCOURSE REPRESENTATION STRUCTURES** take the place of files. Discourse representation structures (DRSs) are in a way one big file card, with information about all of the discourse referents all combined together. For example, the DRS for the discourse in (37) would look as follows:

<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 \( \text{woman}(x) \), or complex, like \( \neg K \) or \( K \Rightarrow K' \), where \( K \) and \( K' \) are both DRSs.

An indefinite adds a new discourse referent to the universe, and subsequent anaphora can update the information associated with that discourse referent. So, spoken out of the blue, a sentence with two indefinites like *a farmer owns a donkey* would give rise to the following DRS:

\[
\begin{array}{c|c}
 x & y \\
\hline
 \text{farmer}(x) & \\
 \text{donkey}(y) & \\
 \text{owns}(x,y) & \\
\end{array}
\]

The same sentence used as the antecedent of a conditional would appear as a DRS contained in a larger DRS, as follows:

\[
\begin{array}{c|c|c}
 x & y & \Rightarrow \\
\hline
 \text{farmer}(x) & \\
 \text{donkey}(y) & \\
 \text{owns}(x,y) & \\
\end{array}
\]

\[
\begin{array}{c}
 \text{beats}(x,y) \\
\end{array}
\]

Informally, a DRS \( K \) is considered to be true in a model \( M \) if there is a way of associating individuals in the universe of \( M \) with the discourse referents of \( K \) so that each of the conditions in \( K \) is verified in \( M \). An EMBEDDING is a function that maps discourse referents to individuals (like an assignment or sequence). The domain of this function will always be some set of discourse referents, but it may or may not include all of the possible discourse referents. In this sense, the function may be a PARTIAL FUNCTION on the set of discourse referents.
Truth in DRT is defined relative to a DRS. A DRS is defined to be true relative to a model if there is an embedding that verifies it in the model. Which embeddings verify a given DRS is determined by semantic clauses for DRSs. But to give an idea, consider the following DRS:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Farmer(x)</td>
<td>Donkey(y)</td>
</tr>
<tr>
<td>Owns(x,y)</td>
<td></td>
</tr>
</tbody>
</table>

A function $g$ verifies this DRS with respect to model $M$ if:

- the domain of $g$ contains at least $x$ and $y$
- according to $M$ it is the case that $g(x)$ is a farmer, $g(y)$ is a donkey, and $g(x)$ owns $g(y)$.

As in predicate logic, we have models $M = \langle D, I \rangle$. $I$ assigns an extension to every predicate (farmer, donkey, owns, etc.). $I(\text{Farmer})$ will be a set of individuals; $I(\text{Owns})$ will correspond to a relation. So $g$ verifies $\text{Farmer}(x)$ with respect to model $M = \langle D, I \rangle$ if and only if $g(x) \in I(\text{Farmer})$. What this means is that an embedding $g$ verifies the DRS for *A farmer owns a donkey* if it assigns $x$ to a farmer, and $y$ to a donkey that the farmer owns.

In general, verification of a DRS is defined as follows:

(39) **Verification of a DRS**

Embedding $g$ verifies DRS $K$ in model $M$ if and only if $g$ verifies every condition in $K$, and the domain of $g$ includes every discourse referent in the universe of $K$.

Whether or not a given embedding $g$ verifies a given condition depends on the nature of the condition. Let us use the notation

$$M, g \models \phi$$

to denote ‘$g$ verifies condition $\phi$ in model $M$’. The rule for deciding whether a given embedding verifies a condition like $\text{Farmer}(x)$, where a predicate applies to an argument, is defined as follows:
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:

\[
\begin{array}{c|c}
  x & y \\
  \hline
  \text{Farmer}(x) & \\
  \text{Donkey}(y) & \\
  \text{Owns}(x,y) & \\
\end{array}
\]

According to the rules that we have set out, this DRS will be true in \( M \) if there is an embedding \( g \) with a domain that includes \( x \) and \( y \), which verifies all three of the conditions, in other words, if there are indeed \( x \) and \( y \) such that \( x \) is a farmer and \( y \) is a donkey and \( x \) owns \( y \).

**Exercise 9.** Under this treatment, indefinites in unembedded sentences like *A farmer owns a donkey* are interpreted essentially as existential quantifiers. Suppose that your friend doesn’t understand why this is so, and explain it to them so that they say ‘Aha!’.

Another kind of atomic condition is equality:

\[
M, g \models x = y \iff g(x) = g(y)
\]

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 *Paul does not own a donkey.*
Intuitively, this should be true if and only if there is no way to assign a value to $x$ such that $x$ is Paul, and there is some individual $y$ such that $y$ is a donkey and $x$ owns $y$. This is defined with the help of some auxiliary notions:

- **Compatibility**
  We say that two functions $f$ and $g$ are **compatible** if they assign the same values to those arguments for which they are both defined. I.e., $f$ and $g$ are compatible if for any $a$ which belongs to the domain of both $f$ and $g$:
  \[ f(a) = g(a) \]

- **Extension**
  $g$ is called an **extension** of $f$ if $g$ is compatible with $f$ and the domain of $g$ includes the domain of $f$.

Thus if $g$ is an extension of $f$ then $f$ and $g$ assign the same values to all arguments for which $f$ is defined, while $g$ may (though it need not) be defined for some additional arguments as well.

Returning to negation:

(42) **Verification of a negated condition**
An embedding function $f$ verifies a condition of the form $\neg K$ with respect to model $M$ iff there is no function $g$ such that:
- $g$ extends $f$
- $g$ verifies $K$
Thus, for example, a function $f$ verifies the negated condition in the DRS for *Paul does not own a donkey* iff:

- $f$ verifies $x = p$, and
- There is no function $g$ such that: (i) $g$ extends $f$, and (ii) $g$ verifies

<table>
<thead>
<tr>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>donkey(y)</td>
</tr>
<tr>
<td>owns(x,y)</td>
</tr>
</tbody>
</table>

This gives us results for negated sentences containing indefinites on par with Russell’s treatment: Just as with negated existentials, a negated sentence containing an indefinite that takes scope under the negation will be true only if there is no object in the model satisfying the relevant description. Furthermore, the fact that the discourse referent is introduced in a DRS that is nested within another DRS, and, as it were, “shielded” from the top level by a negation symbol, gives us the tools to account for the fact that *a donkey* does not license an antecedent in a later sentence. We will not go through how this works here; suffice it to say that the discourse referent is not ACCESSIBLE for subsequent anaphora in this position.

**Exercise 10.** Partially specify a model $M = \langle D, I \rangle$ where *Paul does not own a donkey* is true, by specifying the value of $I$ for the relevant constants. Then give an embedding function $f$ that verifies the negated condition in the DRS for *Paul does not own a donkey* in $M$, and explain why it verifies that condition.

The semantics of conditionals uses the concept of extensions among embedding functions as well.

(43) **Verification of a conditional condition**

An embedding function $f$ verifies a condition of the form
\( K \Rightarrow K' \) with respect to model \( M \) if and only if: For all extensions \( g \) of \( f \) that verify \( K \), there is an extension \( h \) of \( g \) that verifies \( K' \).

The intuitive idea is something like the following: To verify a conditional statement, first consider what kind of embedding would be necessary to verify the antecedent. Now consider whether or not the consequent has to hold, given that embedding.

It turns out that this semantics for conditionals allow for a unified account of indefinites across the full range of uses we have seen. In particular, although unembedded indefinites get an existential interpretation, indefinites acquire universal import in conditionals, and indefinites can bind from antecedent to consequent. Consider the DRS for the donkey sentence:

\[
\begin{array}{|c|c|}
\hline
x & y \\
\hline
\text{farmer}(x) & \\
\text{donkey}(y) & \\
\text{owns}(x,y) & \\
\hline
\end{array}
\Rightarrow
\begin{array}{|c|}
\hline
\text{beats}(x,y) \\
\hline
\end{array}
\]

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 \( M = \langle D, I \rangle \) such that the following statements hold:

\[
\begin{align*}
I(\text{Paul}) &= a \\
I(\text{Farmer}) &= \{a, b, c\} \\
I(\text{Donkey}) &= \{d, e, f\} \\
I(\text{Owns}) &= \{(a, d), (b, e), (b, f)\} \\
I(\text{Beats}) &= \{(a, d), (b, e), (b, f)\}
\end{align*}
\]

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 11.** Let $g$ be the empty embedding $\emptyset$. Using the assumptions about the model given in (45), list all of the embeddings $g$ that verify the antecedent DRS in (44). For each of those embeddings, give an embedding $h$ that verify the consequent.

**Exercise 12.** Change the model specified in (45) so that the condition in (44) is not satisfied, and name the embedding $g$ that verifies the antecedent that does not have an extension $h$ that verifies the consequent.

**Exercise 13.** Draw a DRS for *If a farmer beats a donkey, then he beats a friend of the donkey*, and give a model in which the con-
ditional is (non-trivially) satisfied. Give an example of an embedding \( g \) and an extension \( h \) of \( g \) such that \( g \) verifies the antecedent and \( h \) verifies the consequent.

### 9.6 Compositional DRT

We have not yet touched on how composition works, i.e., how files or DRSs are to be constructed from an LF representation. Both file change semantics and DRT look quite different from the systems we have presented in previous chapters. In this section we show that it is possible to formalize DRT within a version of Montague semantics that is based on classical type logic. One advantage of doing this is that the resulting system can be combined with other parts of the system in this book. Moreover, the formalization allows us to avoid the level of discourse representations that is specific to file change semantics and DRT, and to cut down on special-purpose auxiliary notions involved in interpreting DRT. There are many formalizations that combine DRT and Montague semantics, e.g. Dynamic Montague Grammar (Groe-
nendijk & Stokhof 1990b). The system we present here is based on Compositional DRT or CDRT (Muskens 1995b, 1996). CDRT has the advantage of being based on classical logic, which makes it easy to integrate it with the system developed in the other chapters in this book.

Formally, we will be working in a many-sorted version of the logic in Church (1940). The two-sorted version of this logic was studied in Gallin (1975); it is called two-sorted because it uses two basic types, \( e \) for individuals and \( t \) for truth values. To this we will now add a third basic type, \( r \), which will contain discourse referents and names. Conceptually, individuals are entities of the familiar kind (like kings and cabbages) while discourse referents and names are symbols that encode the focus of our attention.
throughout discourse. *Discourse referents* are introduced by indefinites, while *names* are introduced by proper nouns. For each type-\(e\) constant in our language (john, mary etc.), we assume that the domain also contains a type-\(r\) name \(r_{\text{john}}, r_{\text{mary}}\), etc. In addition to names, we will assume that any model contains either a unlimited supply of discourse referents, or in any case one that is sufficient for the purpose of any discourse. To keep things simple, we will start with models that contain just three discourse referents \(r, r', r''\).

Alongside discourse referents, our language will still make use of variables \(x, y, z, x',\) etc., as in Chapter 3; it is important not to confuse them with discourse referents. In particular, variables can be free or bound by quantifiers and by lambda terms, but discourse referents cannot.

From the three basic types \(e, t, r\), we derive functional types as in Chapter 4. In particular, we will make use of the type \(\langle r, e \rangle\), which is the type of functions from discourse referents to individuals. We will refer to objects of this type as *assignments*. For reasons that will become clear shortly, we will use \(i, j, j', j'',\) etc. and \(o\) as variables over assignments.

We assume that every assignment maps every name to the relevant individual in the domain, so that for any assignment \(i\) we have \(i^j_r = \text{John}, i^m_r = \text{Mary} etc.\) By contrast, discourse referents can be mapped by different assignments to different individuals. Treating names and discourse referents in similar ways and giving them the same type allows us to let pronouns and other anaphoric expressions behave in a uniform way regardless of whether their antecedents are proper names or indefinites.

CDRT assignments are similar to the assignment functions that

---

5Muskens (1996) uses the terms *unspecific discourse referents* for our discourse referents and *specific discourse referents* for our names.

6Since discourse referents and names are objects in the model and since variables can range over objects of any type, we could in principle also introduce variables of type \(r\) that range over them. Here we avoid doing so to reduce confusion and because we will not have a need for such variables.
we introduced in Chapter 3 in that they keep track of which symbols stand for which entities. One can think of an assignment as a register that gets updated throughout the discourse. Expressions that can act as potential antecedents, such as indefinites, update the values of discourse referents in assignments, and expressions such as pronouns and definite descriptions retrieve the values of discourse referents. While each assignment is taken to be immutable (like a book that has been published and whose text cannot be edited anymore), we can simulate the process of making a change to an assignment by finding another assignment that is just like the first in all relevant respects other than the relevant change. This is encapsulated in the following definition:

(46) **Definition**
Let $i$ and $o$ be two assignments and $r$ a discourse referent. We write $i[r]o$ to say that $i$ and $o$ differ at most in the value they assign to $r$ (i.e., either $i$ and $o$ agree on everything except $r$ or they do not differ at all).

There are also differences between assignments in the sense of this chapter and assignment functions as we used them in Chapter 3. Assignments in this chapter are contained in the domain of our models, just like individuals, truth values, predicates, relations, and so on. By contrast, the assignment functions that we used in Chapter 3 are not contained in our models; they are used only as devices for interpreting predicate logic formulas. Another difference is that the assignments in this chapter apply to discourse referents (of type $r$) while the assignment functions in Chapter 3 apply to variables of type $e$.

**The semantics of sentences in CDRT** In Section 9.4 the difference between static and dynamic semantics was summarized as follows: Whereas in static semantics, the meaning of a sentence corresponds to a set of world-assignment pairs, the meaning of a sentence in dynamic semantics corresponds to a binary relation
between world-assignment pairs. Keeping the world constant for simplicity, we can say that the meaning of a sentence (its context-change potential) corresponds to a Schönfinkeled binary relation between assignments. Conceptually, a context-change potential is like a DRS. Formally, context-change potentials have the type \( \langle re, \langle re, t \rangle \rangle \); we will abbreviate this type as \( t \) and we will use the letters \( p \) and \( q \) for variables that range over context-change potentials. We will call the first argument to a context-change potential the input assignment and its second element the output assignment, and we will use the letters \( i \) and \( o \) to symbolize them. By convention, we use the leftmost lambda slot for \( i \) and the second-to-leftmost one for \( o \). CDRT extends this view to every subconstituent down to individual words, so that every lexical entry takes two assignments \( i \) and \( o \) as its arguments in addition to whatever other arguments it applies to. The grammar will be set up so that this property is passed up to larger constituents all the way up to sentences.

For example, consider again the discourse in (37) repeated here with discourse referents added. Following standard practice in dynamic semantics, discourse referents are superscripted in those places where they get introduced into the discourse, and subscripted in those places where they get picked up again. For convenience and following common practice in dynamic semantics, we have assumed that anaphoric links in sentences have already been resolved via coindexing before semantic interpretation takes place. This assumption helps us keep things simple to understand because it lets treat pronouns as essentially denoting discourse referents; is not crucial, and we could, instead let pronouns denote variables over discourse referents (Muskens, 2011).

(47) a. A\(^r\) dog bit a woman\(^r\).
   b. She\(_r\) hit him\(_r\) with a \(_r\)\(^{r''}\) paddle.
   c. It\(_{r''}\) broke in half.
   d. The\(_r\) dog ran away.
The context-change potential of sentence (47a) consists in introducing two discourse referents $r$ and $r'$, and updating the context such that whichever entity $r$ refers to is a dog, and whichever entity $r'$ refers to is a woman that was bitten by $r$. In models where more than one dog and/or more than one woman fits the description, there will more than one way to update the context. This suggests that the context-change potential is properly thought of as a relation between input and output contexts, rather than a function from input to output contexts.

Formally, sentence (47a) denotes the following context-change potential:

\[
\lambda i \lambda o. \exists j. i[r] j \land j[r'] o \\
\land \text{dog}(o(r)) \land \text{woman}(o(r')) \land \text{bite}(o(r), o(r'))
\]

In words, this is (the Schönfinkeled version of) the relation holds between any two assignments $i$ and $o$ just in case they differ at most in what they assign to $r$ and $r'$, and furthermore $o$ maps $r$ to some dog and $r'$ to some woman whom that dog bit.

To keep things readable, from this point onwards for any assignment $j$ and discourse referent $r$, we will write the lookup operation $j(r)$ as $j_r$. Thus the relation above can be written as follows:

\[
\lambda i \lambda o. \exists j. i[r] j \land j[r'] o \\
\land \text{dog}(o_r) \land \text{woman}(o_{r'}) \land \text{bite}(o_r, o_{r'})
\]

In a model where indeed a dog bit a woman, this relation will be nonempty. To take an example at random, in a model that corresponds to World 1 in Section 9.4 in which Pug bit Joan, the following pair of assignments $i^1$ and $o^1$ will stand in the relation (49):

\[
i^1 = \left[\begin{array}{c}
    r & \rightarrow & \text{Homer} \\
    r' & \rightarrow & \text{Homer} \\
    r'' & \rightarrow & \text{Marge}
\end{array}\right] \quad o^1 = \left[\begin{array}{c}
    r & \rightarrow & \text{Pug} \\
    r' & \rightarrow & \text{Joan} \\
    r'' & \rightarrow & \text{Marge}
\end{array}\right]
\]
The values that \( i^1 \) assigns to \( r \) and \( r' \) are irrelevant, and so is the value that both \( i^1 \) and \( o^1 \) assign to \( r'' \). These values have been filled in only for concreteness. Many other assignments than \( i^1 \) and \( o^1 \) stand in the relation denoted by (49). For example, since the values that the input assignment assigns to \( r \) and \( r' \) are irrelevant, \( i^1 \) and \( o^1 \) could be replaced by any other pair of assignments, so long as they map \( r'' \) to the same value as each other and the second assignment still maps \( r \) and \( r' \) to the same values as \( o^1 \) does. This means that the relation (49) will relate any input assignment to at least one output assignment. We will say that a relation that relates \( i \) to some output assignment succeeds on \( i \) (otherwise it fails on \( i \)); thus, the relation (49) succeeds on every input assignment.

The next sentence, (47b), denotes the following context-change potential:

\[
\lambda i \lambda o. i[r'']o \land \text{hit-with}(o_{r'}, o_r, o_{r''}) \land \text{paddle}(o_{r''})
\]

This relation holds between assignments \( i \) and \( o \) just in case they differ at most in what they assign to \( r'' \), and furthermore \( o \) maps \( r'' \) to some paddle which was used by whatever \( o \) assigns to \( r' \) in order to hit whatever \( o \) assigns to \( r \).

What kinds of assignments stand in this relation? Since \( o \) and \( i \) must agree in everything except possibly \( r'' \), they must both assign the same value to \( r \), and likewise for \( r' \). As for \( r'' \), it does not matter what \( i \) assigns it to, but \( o \) must assign it to the right kind of paddle.

For example, consider again a model that is like World 1, where Joan hit Pug with Paddle. Suppose that no other hittings took place. In this model, for two assignments \( i^2 \) and \( o^2 \) to stand in the relation (50) \( i^2 \) must be exactly as below except that \( r'' \) could also be mapped to any other value than Marge; and \( o^2 \) must be exactly as given below.
Because of the constraints it imposes, the relation \[ (50) \] does not succeed on every input assignment. In general, CDRT uses such relations as denotations of sentences that contain unresolved anaphoric dependencies (e.g. unbound pronouns such as \( She_{r'} \) and \( him_{r} \) in \[ (47b) \]).

Typically, the previous discourse will supply input assignments on which such sentences succeed. For example, the pronouns in \[ (47b) \] have their antecedents in the previous sentence \[ (47a) \].

To connect pronouns with their antecedents, we now combine the two denotations \[ (49) \] and \[ (50) \] by an operator called sequencing and written as a semicolon (\( ; \)). This operator is introduced here as a shorthand:

\[
(51) \quad ; =_{\text{def}} \lambda p \lambda q \lambda i \lambda o. \exists j. p(i)(j) \land q(j)(o)
\]

This operator, which is present in many programming languages, takes two context-change potentials \( p \) and \( q \) and combines them to a new one which asserts that some assignment \( j \) can serve as both the output of \( p \) and the input of \( q \). Mathematically, this amounts to composing the relations \( p \) and \( q \); in procedural terms, this amounts to letting the output assignments of \( p \) serve as the input assignments of \( q \). For example, the output assignment \( o_1 \) above is the same as the input assignment \( i^2 \); therefore, \( i_1 \) and \( o^2 \) will stand in the relation denoted by sequencing \[ (49) \] with \[ (50) \]. That relation is the following:

\[
(52) \quad \lambda i \lambda o \exists j \exists j'. i[r]j \land j'[r']j' \\
\land \text{dog}(j'_r) \land \text{woman}(j'_r) \land \text{bite}(j'_r, j'_r) \\
\land j'[r'']o \land \text{hit-with}(o_r, o_r, o_r) \land \text{paddle}(o_r)
\]

In prose and simplifying a bit, this relation holds between two as-
signments $i$ and $o$ just in case $o$ is the result of making minimal changes to $i$ such that $r$, $r'$, and $r''$ are mapped to a dog, a woman that it bit, and a paddle that she hit it with.

**Bridging principles**  Context-change potentials are relations between input assignments and output assignments. But we are used to thinking of sentences as simply being true or false. To know whether a given sentence is true or false in a model, we can convert its context-change potential into a truth value via the following bridging principles. The first bridging principle defines truth and falsity relative to an assignment:

(53)  **Bridging Principle 1**
Let $i$ be an assignment and $\phi$ be a term of type $t$ (i.e. a context-change potential). $\phi$ is true relative to $i$ iff there is an assignment $o$ such that $i[\phi]o$ is true; otherwise $\phi$ is false relative to $i$.

The idea behind this principle is that if we only care whether a sentence is true given its input assignment, and not about whether it provides potential antecedents to subsequent sentences, then it does not matter what output assignments it produces.

For sentences without unresolved anaphoric dependencies, i.e. sentences without pronouns or definite descriptions in them, we can also define truth and falsity simpliciter by universally quantifying over input assignments:

(54)  **Bridging Principle 2**
Let $\phi$ be a term of type $t$ without unresolved anaphoric dependencies. $\phi$ is true iff it is true relative to every input assignment (in the sense of Bridging Principle 1); otherwise it is false.

The idea here is that if a sentence is true in the intuitive sense, then we expect it to remain true no matter what input assignment we present it with.
In combination, the upshot of these two principles is that a context-change potential without unresolved anaphoric dependencies is true just in case it maps every input assignment to some output assignment. For example, according to these principles, the context-change potential in (52) is true just in case for every assignment \( i \) there is an assignment \( o \) that is just like \( i \) except that it maps \( r, r', \) and \( r'' \) to a dog, a woman that it bit, and a paddle that she hit it with. Now suppose that indeed there exist a dog, a woman, and a paddle such that the dog bit the woman and the woman hit the dog with the paddle. Then for any input assignment \( i \) such an output assignment \( o \) can be obtained by changing \( i \) as needed so that it maps \( r \) to the dog, \( r' \) to the woman, and \( r'' \) to the paddle.

The reason that Bridging Principle 2 is restricted to sentences that do not have unresolved anaphoric dependencies in order to avoid collapsing the truth conditions of pronouns and corresponding universals. Without this constraint, a sentence like (55a) would have the same truth conditions as Heraclitus’ famous aphorism in (55b).

\[(55)\quad a. \quad \text{It}_r \text{ is in flux.}\]
\[b. \quad \text{Everything is in flux.}\]

This is because (55a) is true relative to any input assignment that maps \( r \) to something in flux. Suppose now that everything is in flux; then, and only then, every assignment whatsoever will map \( r \) to something in flux. Suppose instead that some things are in flux and others aren’t; in that case, some assignments will map \( r \) to something in flux, while others will not. Accordingly, (55a) will be true (in the sense of Bridging Principle 1) with respect to some assignments but not others.

There is an intuitive connection between sentences with unresolved anaphoric dependencies in CDRT and formulas with free variables in predicate logic. Both can be true with respect to some assignments and false with respect to others. More generally, the
input assignments in CDRT play an analogous role to the assignment functions in predicate logic.

**Lexical entries for CDRT** One of the advantages of using the lambda calculus to express context-change potentials is that we can now rely on it to generate them compositionally in the usual way. To do this, we equip each lexical entry with two extra slots $\lambda i$ and $\lambda o$. For those lexical entries that do not introduce new discourse referents, we add a conjunct that requires $i = o$; otherwise almost any pair of assignments could serve as input and output and anaphoric information would be lost. For example, here are some nouns and intransitive verbs:

(56) a. $\text{woman} \sim \lambda x \lambda i \lambda o. i = o \land \text{woman}(x)$  
b. $\text{dog} \sim \lambda x \lambda i \lambda o. i = o \land \text{dog}(x)$  
c. $\text{run-away} \sim \lambda x \lambda i \lambda o. i = o \land \text{run-away}(x)$

The type of these entries is $\langle e, t \rangle$ (recall that we use $t$ to abbreviate $\langle (r, e), \langle (r, e), t \rangle \rangle$, the type of context-change potentials).

Proper nouns simply denote the relevant individuals, as usual:

(57) $\text{John} \sim \text{john}$

Indefinites introduce discourse referents $r$ by operating on the input assignment $i$ and by using an intermediate assignment $j$ that is constrained to differ from $i$ at most in $r$. They also take a restrictor $R$ and a nuclear scope $N$, both of type $\langle e, t \rangle$, pass the value of $r$ according to $j$ to $R$ and $N$ and link them up via sequencing.

(58) $a^r \sim \lambda R \lambda N \lambda i \lambda o \exists j. i[r] j \land (R(j_r); N(j_r))(j)(o)$

For the sake of readability, from here on we will write $\phi(i)(o)$ as $i[\phi]o$, for any formula $\phi$ of type $t$; thus this above simplifies as follows:

(59) $a^r \sim \lambda R \lambda N \lambda i \lambda o \exists j. i[r] j \land j[R(j_r); N(j_r)]o$
We can also spell out the sequencing shorthand to make things clearer:

\[(60)\quad d' \sim \lambda R \lambda N \lambda i \lambda o \exists j .
\]
\[
i[r] j \wedge \exists j'. j[R(j_r)] j' \wedge j'[N(j_r)] o
\]

Using this entry and two instances of function application, sentence \(61a\) evaluates to \(61b\) which is equivalent to \(61c\) due to the equivalences between assignments:

\[(61)\]
\[
a. \quad A^r \text{ dog ran away.}
b. \quad \lambda i \lambda o . i[r] j \wedge \exists j'. \text{dog}(j_r) \wedge j = j'
\]
\[
\wedge \text{run-away}(j_r) \wedge j' = o
\]
\[
c. \quad \lambda i \lambda o . i[r] o \wedge \text{dog}(o_r) \wedge \text{run-away}(o_r)
\]

That \(61b\) is so much more complicated than \(61c\) is due to the fact that neither the restrictor nor the nuclear scope of the indefinite \(a\) in \(61a\) happen to contain any indefinites or anything else that introduces discourse referents. In general, though, this is not always the case; and this is also the reason for the sequencing operator in \(59\). The point of sequencing \(R\) and \(N\) is to preserve any anaphoric links from within \(R\) into \(N\), such as the link between \(a\) donkey and \(it\) in examples like the following:

\[(62)\quad A^r [_{Restr} \text{ farmer who had a}^r \text{ donkey}] [_{Nucl} \text{ beat it}_r].\]

Before we get to such examples, we will build up the rest of our lexicon as we need it for our toy discourse. Consider first pronouns. We will ignore gender and case features and simply treat them as devices that query an input assignment for the value of the discourse referent they are indexed with. We could let the pronoun just return this value, but this would prevent them from combining with predicates such as verb phrases; such predicates expect an individual, not a relation between assignments and individuals. To remedy this, we let the pronoun take its predicate as an additional argument; this is called Montague-lifting the pronoun.
Here, \( P \) is of type \((e, t)\); thus, the type of any pronoun is \((e, t)\). In general, all noun phrases in CDRT are of this type.

\[
\text{(63) } he_r/him_r/she_r/her_r/it_r \sim \lambda P \lambda i \lambda o. i = o \wedge i[P(i_r)]o
\]

For example, \((64a)\) denotes \((64b)\): 

\[
\begin{align*}
\text{(64) } & \quad \text{a. It}_r \text{ ran away.} \\
& \quad \text{b. } \lambda i \lambda o. i = o \wedge \text{run-away}(i_r)
\end{align*}
\]

Pronouns can also be indexed with names rather than discourse referents. Recall that our model contains names like \( r^\text{john} \) that every assignment maps to the relevant individual, so that for any assignment \( i \) we have \( i^r_{\text{john}} = \text{John} \). This means that \((65a)\) is equivalent to \((65b)\):

\[
\begin{align*}
\text{(65) } & \quad \text{a. } he^r_{\text{john}} \sim \lambda P \lambda i \lambda o. i = o \wedge i[P(i_r)]o \\
& \quad \text{b. } he^r_{\text{john}} \sim \lambda P \lambda i \lambda o. i = o \wedge i[P(\text{john})]o
\end{align*}
\]

Turning now to definite descriptions, we assume following Heim \(1982b\) that they behave just as anaphoric pronouns do, except that they come with additional descriptive content. Formally, definite determiners combine with a restrictor and a nuclear scope, which are both applied to the entity they refer to.

\[
\text{(66) } \quad \text{the}_r \sim \lambda R \lambda N \lambda i \lambda o \exists j. i[R(j_r)] \wedge j[N(j_d)]o
\]

Consider now a transitive verb such as \( \text{bite} \). Following the same reasoning as before, we arrive at the following lexical entry:

\[
\text{(67) Preliminary entry } \\
\text{bite} \sim \lambda y \lambda x \lambda i \lambda o. i = o \wedge \text{bite}(x, y)
\]

This entry cannot combine with noun phrases, since they are of type \((\langle e, t \rangle, t)\) rather than \(e\). To avoid this type mismatch, we apply type shifting to the lexical entries of transitive and ditransitive verbs (for convenience, \( \text{hit with} \) is treated as if it was a ditransitive verb). To do so, we use the Hendriks schema presented in
Section 7.3.2 to generate an Object Raising rule that is adapted for the dynamic setting. This results in the following entry:

(68) Final entry
\[
\text{bite} \sim \lambda Q \lambda x \lambda i \lambda o \cdot Q(\lambda y.i = o \land \text{bite}(x,y))
\]

**Exercise 14.** Derive the adapted Object Raising rule that is appropriate for the CDRT setting, and derive (68) from (67).

In the same way, we can use Hendriks’ schema to lift the direct and indirect objects of ditransitive verbs:

(69) Preliminary entry
\[
\text{hit-with} \sim \lambda y \lambda z \lambda x \lambda i \lambda o \cdot i = o \land \text{hit-with}(x,y,z))
\]

(70) Final entry
\[
\text{hit-with} \sim \lambda Q' \lambda Q \lambda x \lambda i \lambda o.
\]
\[
Q'(\lambda y.Q(\lambda z.i = o \land \text{hit-with}(x,y,z)))
\]

Using these entries, we can generate context change potentials for the sentences in (47). We have already seen the context-change potentials for (47a) and (47b) in (49) and (50). The one for (47c) is analogous to the one in (64b) and the one for (47d) is similar.

**Exercise 15.** Using the appropriate CDRT lexical entries, give a compositional derivation of the context change potential of Sentence (47d). Show the details of the derivation. Use equivalences between assignments to simplify the result as much as possible in the same manner shown in (61b) and (61c).
**Exercise 16.** We have seen how the context change potentials of sentences (47a) and (47b) can be combined using sequencing, as well as some examples of assignments that can serve as inputs and outputs to each of these sentences, with the output of (47a) serving as input to (47b). Do the same for the transitions from (47b) to (47c), and from (47c) to (47d). Using repeated sequencing, produce a context-change potential for the entire discourse. Paraphrase its truth conditions. Explain how anaphoric dependencies are realized and preserved.

An advantage of CDRT is that negation and conditionals do not require us to define any special composition rules. We can rely on function application for these operators just as for any other lexical entry. The following entry for *not* assumes the VP-internal subject hypothesis:

\[
not \sim \lambda p \lambda i \lambda o. i = o \land \neg \exists j. i[p]j
\]

**Exercise 17.** Modify this entry so that it takes a subject of type \(\langle \langle e, t \rangle, t \rangle\) along with a VP that expects a subject of that type.

This entry limits the lifespan of discourse referents in its scope so that they are no longer available for pronouns in subsequent sentences to pick up:

\[
Paul \text{ does not own a}^r \text{ donkey. \#It}^r \text{ is grey.}
\]

Using analogous lexical entries to the ones we have already seen, we combine the transitive verb with the indefinite object and get the following:

\[
\text{own a}^r \text{ donkey} \\
\sim \lambda x \lambda i \lambda o. \exists j. i[r] j \land \text{donkey}(j_r) \land \text{own}(x, j_r)
\]
After combining with the subject, we get:

\[(74) \quad Paul\; owns\; a^r\; donkey\]
\[\sim \lambda i \lambda o . . i[r] o \land donkey(o_r) \land own(paul, o_r)\]

This context-change potential relates any two assignments \(i\) and \(o\) just in case they differ at most in \(r\) and \(r'\), in such a way that \(o\) maps \(r\) to a donkey that Paul owns.

The bridging principles in (53) and (54) have the effect that this is true just in case there exist an assignment \(i\) and an assignment \(o\) that stand in this relation.

Negation now applies and converts this into a context-change potential that requires \(i\) and \(o\) to be identical, and furthermore ensures that there is no assignment that is like \(i\) aside from mapping \(r\) to a donkey Paul owns:

\[(75) \quad not(Paul\; owns\; a^r\; donkey)\]
\[\sim \lambda i \lambda o . i = o \land -\exists j . i[r] j \land donkey(j_r) \land own(paul, j_r)\]

The bridging principles have the effect that this is true just in case there is an assignment \(i\) such that for no assignment \(o\) is it the case that \(i\) differs from \(o\) at most in that \(o\) maps \(r\) to a donkey Paul owns. That is to say, there is an assignment \(i\) such that every assignment \(o\) differs from \(i\) in more than the fact that \(o\) maps \(r\) to a donkey Paul owns.

This chapter has only given a taste of dynamic semantics, enough to show that it has the power to deal smoothly with the apparently variable force of indefinites. Geurts & Beaver (2011) provide a more thorough overview, including more on the notion of ‘accessibility’, which constrains the ‘lifespan’ of discourse referents. The interested student is encouraged to start there and work backwards from the references cited there.
10  |  Coordination and Plurals

10.1  Coordination

Let us now consider coordination in more detail. We may include sentences with *and* and *or* among the well-formed expressions of our language by extending our syntax and lexicon as follows:

(1)    **Syntax**
S   →   S 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*.
Sentences are not the only kinds of expressions that can be coordinated, though. Here are a few examples:

(5)  a. Somebody smokes and drinks. (VP and VP)
    b. No man and no woman arrived. (DP and DP)
    c. Susan caught and ate the fish. (V and V)

It is clear that we need to extend our grammar. Since these examples do not cover all the possibilities, it will not do to introduce fixes to the syntax and semantics one at a time. Instead, we need to formulate a general pattern and then extend our syntax and semantics according to it.

How shall we analyze the semantics of coordination? An early style of analysis consisted in analyzing all coordinations as underlyingly sentential, even those of constituents other than sen-
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 1.** For each of the two sentence pairs above, establish that they are not equivalent by describing a scenario in which one of them is true and the other one is false.

   Luckily, it is also possible to design a grammar in which coordinated constituents are directly generated syntactically, and directly interpreted semantically. We can extend the syntax by pairs of rules of the following kind, one pair for each category:

(9) **Syntax**
    \[
    X \rightarrow X \text{JP} \\
    \text{JP} \rightarrow J X
    \]
where $X \in \{S, VP, DP, V, \ldots\}$

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*.

(11) $\exists x. \text{Smokes}(x) \land \text{Drinks}(x)$

What about coordinations of transitive verbs, as in *Marge loves and hates Homer*? Assuming that transitive verbs translate to ex-
pressions of type \( \langle e, \langle e, t \rangle \rangle \), that is, (Schönfinkelized) binary relations, the version of \textit{and} that should be used to coordinate them should take two binary relations and return a new binary relation. The following entries will do that trick.

\begin{align*}
(12) & \quad \text{a. } \text{and} \mapsto \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \land R'(y)(x) \\
& \quad \text{b. } \text{or} \mapsto \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \lor R'(y)(x)
\end{align*}

Given \textit{loves} and \textit{hates}, these lexical entries will produce a new relation, ‘loves and hates’.

**Exercise 2.** Using the lexical entry for \textit{and} above, draw the tree for \textit{Susan caught and ate the fish}.

Noun phrase coordination can be approached in the same way. Let us first look at conjunctions of quantifiers:

\begin{align*}
(13) & \quad \text{a. Every man and every woman arrived.} \\
& \quad \forall x. [\text{Man}(x) \rightarrow \text{Arrived}(x)] \land \forall x. [\text{Woman}(x) \rightarrow \text{Arrived}(x)] \\
& \quad \text{b. A man or a woman arrived.} \\
& \quad \exists x. [\text{Man}(x) \land \text{Arrived}(x)] \lor \exists x. [\text{Woman}(x) \land \text{Arrived}(x)]
\end{align*}

Since quantifiers have a higher type, they take verb phrases as arguments. This makes the entries for \textit{and} and \textit{or} very similar to their VP-coordinating counterparts:

\begin{align*}
(14) & \quad \text{a. } \text{and}_{DP} \mapsto \lambda Q' \lambda Q \lambda P. Q(P) \land Q'(P) \\
& \quad \text{b. } \text{or}_{DP} \mapsto \lambda Q' \lambda Q \lambda P. Q(P) \lor Q'(P)
\end{align*}

**Exercise 3.** Using the lexical entries above, draw the trees for \textit{Every man and every woman arrived} and \textit{A man or a woman arrived}.

In all of the examples so far, the two constituents being coordinated were of the same semantic type. That is not always the
case. As the following example shows, a type-$e$ noun phrase like John can be coordinated with a type-$(e, t, t)$ noun phrase.

(15) John and every woman arrived.

The translation we should obtain for this sentence is as follows:

$$\text{Arrived}(j) \land \left[ \forall x. \text{Woman}(x) \rightarrow \text{Arrived}(x) \right]$$

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 4. Entity-to-quantifier shift**

If $\alpha \sim \alpha'$, where $\alpha'$ is of type $e$, then $\alpha$ can also be translated as follows:

$$\lambda P. P(\alpha')$$

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 $(e, t, t)$. In a sentence like John arrived, it can take the verb phrase as an argument. In a sentence like John and every woman arrived, we are able to conjoin it with every woman using the entry and$_{DP}$.

**Exercise 4.** Draw the tree for John and every woman arrived and derive a semantic interpretation for it compositionally.
Exercise 5. Draw a tree for *Homer and Marge smoke* and give a derivation that results in:

\[ \text{Smoke}(h) \land \text{Smoke}(m) \]

You will need to apply the type shifter once on each conjunct.

Exercise 6. 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 \( t \).

\[
\langle \land \rangle_{\langle \tau, \langle \tau, \tau \rangle \rangle} = \begin{cases} 
\lambda q \lambda p. p \land q & \text{if } \tau = t \\
\lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1} \cdot \langle \land \rangle_{\langle \sigma_2, \langle \sigma_2, \sigma_2 \rangle \rangle} (X(Z)) (Y(Z)) & \text{if } \tau = \langle \sigma_1, \sigma_2 \rangle
\end{cases}
\]

\[
\langle \lor \rangle_{\langle \tau, \langle \tau, \tau \rangle \rangle} = \begin{cases} 
\lambda q \lambda p. p \lor q & \text{if } \tau = t \\
\lambda X_\tau \lambda Y_\tau \lambda Z_{\sigma_1} \cdot \langle \lor \rangle_{\langle \sigma_2, \langle \sigma_2, \sigma_2 \rangle \rangle} (X(Z)) (Y(Z)) & \text{if } \tau = \langle \sigma_1, \sigma_2 \rangle
\end{cases}
\]

(For more details on this approach, see for example [Partee & Rooth 1983] and [Winter 2001b].)

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:
Coordination and Plurals

(18) $\lambda X_{(t, t)} \lambda Y_{(t, t)} \lambda Z_{(t, t)} \cdot \langle \text{and} \rangle_{(t, t)} (X(Z)) (Y(Z))$

To resolve $\langle \text{and} \rangle_{(t, t)}$, we apply Definition (16) once more. This time, $\tau = t$, so the result is simply logical conjunction:

(19) $\text{and}_{(t, t)} \sim \lambda q \lambda p \cdot p \land q$

We plug this into the previous line and get the final result:

(20) $\lambda X_{(t, t)} \lambda Y_{(t, t)} \lambda Z_{(t, t)} \cdot Y(Z) \land X(Z)$

This is indeed equivalent our entry for DP-coordinating and in (14a). The entry will only work if both DPs are of type $\langle (e, t), t \rangle$. If necessary, one or both DPs may need to be lifted into that type first by applying the type shifter above.

Your task: Show how the schema can be applied to VP-coordination.

10.2 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. Consider the following example.

(21) Homer and Marge are a happy couple. There is no obvious way to formulate the truth conditions of (21) using logical conjunction. It cannot be represented as:

\[
\text{Happy-couple}(h) \land \text{Happy-couple}(m)
\]

since this would entail Happy-couple(h) as well as Happy-couple(m). In other words, it would have the entailments that Homer is a happy couple and that Marge is a happy couple. These
are obviously nonsensical because a singular individual can't be a couple. Only two people can form a happy couple. Predicates like be a couple are called collective. They apply to collections of individuals directly, without applying to those individuals. In this sentence, then, the word and does not seem to amount to logical conjunction but to the formation of a collection, in this case, the “collective individual” Homer-and-Marge.

Another example of collective predication was given by Link (1983a), at the beginning of his paper. He writes:

The weekly Magazine of the German newspaper Frankfurter Allgemeine Zeitung regularly issues Marcel Proust’s famous questionnaire which is answered each time by a different personality of West German public life. One of those recently questioned was Rudolf Augstein, editor of Der Spiegel; his reply to the question: “Which property do you appreciate most with your friends?” was... “that they are very few”.

Clearly, this is not a property of any one of Augstein’s friends; yet, even apart from the esprit it was designed to display the answer has a straightforward interpretation. The phrase... predicates something collectively of a group of objects, here: Augstein’s friends.

To talk about such collections, we need to extend our formal setup. On the semantic side, we will add collections of individuals to our model. You might suspect that we would represent these collections as sets, so that 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 (1983b). Another reason
is that set theory makes formal distinctions that turn out not to be needed in mereology. Where set theory is founded on two relations (\(\in\) and \(\subseteq\)), mereology collapses them into one, the parthood relation. This relation holds both between Homer and Homer-and-Marge (where in set theory, we would use \(\in\)), 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 are a happy couple* can be represented as \(\text{Happy-couple}(h \oplus m)\). Since the sum \(h \oplus m\) is of type \(e\), the type of the VP is \(\langle e, t \rangle\) 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.*

In mereology, the domain can be organized into an algebraic structure. An algebraic structure is essentially a set with a binary operation (in this case, \(\oplus\)) defined on it. Figure 10.1 illustrates such a structure. The circles stand for the individuals Tom, Dick, and Harry, and for the sums that are built up from them. We will use the word **INDIVIDUAL** to range over all the circles in this structure. We will refer to Tom, Dick, and Harry, as **ATOMIC INDIVIDUALS**; the other circles stand for individuals which are not atomic. The lines between the circles stand for the parthood relations that hold between the various individuals. We will assume that parthood is reflexive, transitive, and antisymmetric, or as it is called in mathematics, a “partial order”. Reflexivity means that everything is part of itself. (This may not be intuitive but it is a mere formal convenience, and it can be eliminated by defining a distinct notion of proper parthood: \(a\) is a proper part of \(b\) just in case \(a\) is
both part of and distinct from \( b \).) Transitivity means that if \( a \) is part of \( b \) and \( b \) is part of \( c \), then \( a \) is also part of \( c \). For example, according to Figure 10.1, \( t \) is part of \( t \oplus d \), 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 10.1: An algebraic structure

![Diagram of algebraic structure](image)

The branch of formal semantics that uses algebraic structures and parthood relations to model various phenomena is known as algebraic semantics. The fundamental assumption in algebraic semantics is that any nonempty set of things of the same sort (for example individuals or events) has one and exactly one sum. So far, we have only considered one sort, namely individuals (type \( e \)). We will assume that all individuals, including sums, will be of type \( e \). To express that the atomic individual Tom is part of the sum individual Tom-and-Dick, we will write \( t \leq t \oplus d \). In structures like the one depicted in Figure 10.1, the sum of any nonempty set of individuals \( P \) is always the lowest individual that sits above every element of \( P \). (As we will see later, this corresponds to the mathe-
matical notion of “least upper bound”). For example, if $P$ consists of the two atomic individuals $t$ and $d$, then the lowest individuals that sits above these two is $t \oplus d$. Sometimes the sum of $P$ can be a member of $P$. For example, if $P$ consists of $t$ and $t \oplus d$, then its sum is $t \oplus d$ again. And if $P$ consists of just one individual, such as $h$, then its sum is that individual itself.

10.3 The plural

10.3.1 Algebraic closure

Coordinations of proper nouns are not the only way to talk about sums:

(22)  
a. Tom, Dick and Harry met.
b. Some boys met.
c. Three boys met.
d. The boys met.

In each of these three sentences, the collective predicate $met$ applies to a sum $x$. Only (22a) fully specifies the parts of that sum, while (22b) and (22d) described it partially. That is, we know that they are all boys, but we don’t know who they are. What is the meaning of the noun $boys$? One way to describe it is in terms of the conditions it imposes on $x$, namely, $boys$ requires it to be the sum of 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 1983b]):

(23)  
**Definition: Algebraic closure**

The algebraic closure $^*P$ of a set $P$ is the set that contains
any sum of any nonempty subset of $P$.

The most straightforward way to implement this idea is to identify the meaning of the plural morpheme with the “star operator”:

\[(24) \quad -s \rightarrow \lambda P. \ast P\]

For example, suppose that we are in a model with just three boys, Tom, Dick and Harry. Then the denotation of the noun *boy* might be modeled as \{t, d, h\}. The denotation of the noun *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 *Tom are boys*, and the sentence *Some doctors are in the room* is false if only one doctor is in the room. And indeed, Link himself proposed excluding them. But this leads to a different problem: It makes *boys* essentially synonymous with *two or more boys*. But *No doctors are in the room* is not synonymous with *No two or more doctors are in the room*. Consider the case where a single doctor is in the room. Here only one of the two sentences is true. For this reason we will continue to use \[(24)\] as the meaning of the plural, and rule out *Tom are boys* on pragmatic grounds. That is, *boys* literally means *one or more boys*, and the singular is a special case of the plural. *Boy* and *boys* are in competition, and the singular form blocks the plural form because it is more specific [Sauerland et al. 2005; Spector 2007].

Link gave plural individuals the status of first-class citizens in the logical representation of natural language. This allowed him to represent collective predicates like *meet* as predicates that apply directly to sum individuals:

\[(25)\]

a. Tom (and) Dick and Harry met. $\sim$ Meet(t $\oplus$ d $\oplus$ h)

b. Some boys met.$\sim$ $\exists x. *Boy(x) \land Meet(x)$

As seen in \[25a\] Link represented sentential conjunction in a different way than noun phrase conjunction. This has the conse-
sequence that even the translations of equivalent sentences can look very different:

\[(26) \begin{align*}
a. & \quad \text{Tom is a boy and Dick is a boy.} & \Rightarrow & \text{Boy}(t) \land \text{Boy}(d) \\
& \quad \text{b. Tom and Dick are boys.} & \Rightarrow & *\text{Boy}(t \oplus d)
\end{align*}\]

**Exercise 8.** Draw trees for the sentences in (25) and (26b), 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 (25) and (26b).

### 10.3.2 Plural definite descriptions

Now, supposing that *boys* denotes the set of boy-pluralities, what does *the boys* denote? If we translate *the boys* as:

\[\iota x. *\text{Boy}(x)\]

then we will have a presupposition failure as long as there is more than one boy, because more than one individual will satisfy the predicate \(*\text{Boy}(x)\).\(^1\) How shall we remedy this problem?

One possible solution is to give a different kind of analysis for plural *the*, where it refers to the sum of the individuals that satisfy that predicate given by the noun, rather than the unique individual that satisfies it. The sum operator is usually written with a \(\sigma\) (the Greek letter ‘sigma’), following Link (1987). It is defined as follows:

\[(27) \quad \sigma x. P(x) \text{ is defined as:} \quad \iota x. P(x) \land \forall y[\iota x. *P(y) \rightarrow y \leq x] \]

\(^1\)This was pointed out by Sharvy (1980).
For example, $\sigma x.\text{boy}(x)$ denotes the sum of all of the boys.

The plural definite article can then be treated as denoting this sum operator:

\[(28) \quad \text{the}_{\text{SUM}} \sim \lambda P. \sigma x. P(x)\]

Later we will consider a different version of \textit{the}; the subscript \text{SUM} is there to distinguish this version of \textit{the} from the other one we will consider. Combined with \textit{boys}, this yields:

\[(29) \quad \text{the}_{\text{SUM}} \text{ boys} \sim \sigma x. *\text{Boy}(x)\]

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 10.1, the sum of this set, and therefore the denotation of \textit{the boys}, is just $t \oplus d \oplus h$. This is exactly what we want.

But in other cases, such as \textit{the boy} and \textit{the two boys}, we run into a problem. To see this, let us first establish some assumptions about how phrases like \textit{two boys} are interpreted. Suppose that \textit{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 \textit{boys} via Predicate Modification:

\[(30) \quad \text{two} \sim \lambda x. \text{Card}(x) = 2\]

Then \textit{two boys} will translate as follows:

\[(31) \quad \text{two boys} \sim \lambda x. \text{Card}(x) = 2 \land *\text{Boy}(x)\]

In our model, the set denoted by \textit{two boys} is $\{t \oplus d, t \oplus h, d \oplus h\}$.

\[\textbf{Exercise 9}.\text{ In order to deal with sentences like } Two\text{ boys met, we can assume that there is a silent determiner with the semantics of a generalized existential quantifier:}\]

$$\emptyset_D \sim \lambda P \lambda P'. \exists x. [P(x) \land P'(x)]$$
Give a derivation for *Two boys met* using this assumption. Don’t forget to include the silent determiner in the tree diagram.

Now, what about *the two boys*? If we use \( \text{the}_{\text{SUM}} \) from above, then we will get the sum of the two-boy pluralities. As a glance at Figure 10.1 will confirm, this sum is \( t \oplus d \oplus h \). So we end up with the rather odd prediction that *the two boys* refers to this sum!

**Exercise 10.** Translate \( \text{The}_{\text{SUM}} \) *boys met* and \( \text{The}_{\text{SUM}} \) *two boys met.*

**Exercise 11.** In the model where the boys are Tom, Dick, and Harry, what (if anything) do the expressions \( \text{the}_{\text{SUM}} \) *boy*, \( \text{the}_{\text{SUM}} \) *boys*, \( \text{the}_{\text{SUM}} \) *two boys* and \( \text{the}_{\text{SUM}} \) *three boys* denote? In each case, explain which presupposition arises and whether it is satisfied.

Which of these cases does this theory of plural *the* make the correct predictions for?

Intuitively, *the two boys* should give rise to a presupposition failure, because there are three boys in our model. We must build a source of presupposition failure into our 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):

\[
\text{the}_{\text{SUPR}} \sim \lambda P \forall x[P(x) \land \forall y[P(y) \rightarrow y \leq x]]
\]

We call it this because under this theory, *the* denotes the supremum of the \( P \)'s: the \( P \) that contains all other \( P \)'s.

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 (32) picks out the only member of that singleton. In all other models, the $\iota$ operator will not be defined.

**Exercise 12.** Translate $\text{The}_{\text{SUPR}}$ boys met and $\text{The}_{\text{SUPR}}$ two boys met. This exercise can be solved in the Lambda Calculator.

**Exercise 13.** In the model where the boys are Tom, Dick, and Harry, what (if anything) do the expressions $\text{the}_{\text{SUPR}}$ boy, $\text{the}_{\text{SUPR}}$ boys, $\text{the}_{\text{SUPR}}$ two boys and $\text{the}_{\text{SUPR}}$ three boys denote? In each case, explain which presupposition arises and whether it is satisfied.

Which of these cases does this theory of plural *the* make the correct predictions for?

### 10.4 Cumulative readings

So far, we have seen three kinds of predicates that apply to sums: plural nouns like *boys*, collective predicates like *met*, pluralized distributive predicates like *arrived-∅*. All these are one-place predicates. Sums can also be related by two-place predicates, as in the following sentences:

\[(33)\]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>The men in the room are married to the women across the hall. (Kroch 1974)</td>
</tr>
<tr>
<td>b.</td>
<td>600 Dutch firms use 5000 American supercomputers. (adapted from Scha 1981)</td>
</tr>
<tr>
<td>c.</td>
<td>Tom, Dick and Harry (between them) own (a total of) four gadgets.</td>
</tr>
</tbody>
</table>

Let us take a closer look at the ways the plural entities in these
sentences are related. Sentence (33a) is true in a scenario where each of the men in the room is married to one of the women across the hall, and each of the women is married to one of the men. Sentence (33b) (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 (33c) 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 *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):

(34) **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.

(35) **Definition: Tuple of an $n$-place predicate**  
For a given $n$-place relation $R$, a tuple of $R$ is any $n$-tuple $(x_1, x_2, \ldots, x_n)$ such that $R(x_1)(x_2)\ldots(x_n)$.

(36) **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 tu-
Coordination and Plurals

Examples of $R$. We write $^* R(a,b)$ for $^* R(\langle a,b \rangle)$.

We can then represent cumulative readings by using the algebraic closure of transitive verbs:

(37) Tom, Dick and Harry own four gadgets. $\forall$

$$\exists x_1. ^* \text{gadget}(x) \land \text{card}(x) = 4 \land ^* \text{own}(t \oplus d \oplus h, x)$$

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 $\langle t \oplus d \oplus h, g_1 \oplus g_2 \oplus g_3 \oplus g_4 \rangle$.

10.5 Formal mereology

Intuitively, the sum of some things is that which you get when you put them together. For many purposes, this rough intuition along with a quick glance at diagrams like the one in Figure[10.1] is sufficient. But if we want to prove that the logical representation of one sentence entails that of another sentence, and if these representations involve parthood or sums, then we need a precise framework in which we can substantiate our intuitions. For example, if we want to prove that ((26b)) logically follows from ((26a)), 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.

(38) **Axiom: Transitivity**
Any part of any part of a thing is also part of that thing.

(39) **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.

(40) **Definition: Disjointness**
If two things do not have any part in common, we say that they are disjoint.

(41) **Axiom: Weak supplementation**
Any proper part of any thing is disjoint from one of the parts of that thing.

(42) **Definition: Upper bound**
Something of which everything in P is part is called an upper bound of P.

(43) **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.

(44) **Axiom: Existence of sums**
Every nonempty P has a sum.

(45) **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:

(46) **Theorem: Reflexivity** Everything is part of itself. (TODO: prove this.)

(47) **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 (41), 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 (38) \( c \) is part of \( a \). By Theorem (46), \( a \) is also part of \( a \). So \( a \) and \( c \) have a part in common, namely \( a \), and cannot be disjoint. Contradiction.)

(48) **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 (43), 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 (47).)

(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 (48) 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 (23), \( ^*\text{boy} \) 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.

10.6 A formal fragment

Finally, we need to extend the syntax and semantics of our logic in order to adapt it to the new entities and relations we have added. The syntax is defined as in three-valued type logic ($L_\lambda$ with three truth values as in Chapter 8), plus the following additions:

10.6.1 Logic syntax

We add the following primitive symbol to our syntax.

1. **Parthood** If $\alpha$ 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 $\langle e, t \rangle$, we write $\oplus \phi$ (read as: “the sum of $\phi$”) for the expression $[\lambda x.[\forall y.[\phi(y) \to y \leq x]] \land [\forall z.[\forall z'.[\phi(z') \to z' \leq z]] \to x \leq z]]$. (That is, $\oplus \phi$ denotes the least upper bound, or sum, of $\phi$ – see Definition (43). By Axiom (48), 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 $\oplus[\lambda x.x = \alpha \lor x = \beta]$.

3. An expression of the form $[[\alpha \oplus \beta] \oplus \gamma]$ or $[\alpha \oplus [\beta \oplus \gamma]]$ can be simplified to $[\alpha \oplus \beta \oplus \gamma]$. 
10.6.2 Logic semantics

Expressions are interpreted with respect to both:

- a model \( M = (D, I, \leq) \) where \( D \) and \( I \) are defined as usual and \( \leq \) is the parthood relation over individuals that obeys the conditions listed above,
- an assignment \( g \) defined as usual.

For every well-formed expression \( \alpha \), \( [\alpha]^{M,g} \) is defined recursively as usual. We add the following rule:

\[(50) \text{ Parthood} \]

If \( \alpha \) and \( \beta \) are expressions of type \( e \), then \( [\alpha \leq \beta] = 1 \) if \( [\alpha] \leq [\beta] \), otherwise \( [\alpha \leq \beta] = 0 \).

10.6.3 English syntax

Syntax rules. We add the following rules for coordination:

\[(51) \text{ Syntax} \]

\[
X \rightarrow X \text{ JP} \\
\text{JP} \rightarrow J X \\
\text{where } X \in \{S, VP, DP, V, \ldots\}
\]

In addition, we add the following rule for the plural:

\[
N \rightarrow N \text{ Pl}
\]

Lexicon. Lexical items are associated with syntactic categories as follows:

\[
\begin{align*}
D: & \quad \emptyset_D \\
A: & \quad \text{two, three etc.} \\
J: & \quad \text{and, or} \\
\text{Pl}: & \quad -s \\
V: & \quad \text{met, own}
\end{align*}
\]
10.6.4 Translations

Type \( (e, t) \):

1. \( \text{smokes} \sim \lambda x. \ast \text{smoke}(x) \)
2. \( \text{drinks} \sim \lambda x. \ast \text{drink}(x) \)
3. \( \text{two} \sim \lambda x. \text{card}(x) = 2 \)

Type \( (e, \langle e, t \rangle) \):

1. \( \text{caught} \sim \lambda y \lambda x. \ast \text{catch}(y)(x) \)
2. \( \text{ate} \sim \lambda y \lambda x. \ast \text{eat}(y)(x) \)
3. \( \text{own} \sim \lambda y \lambda x. \ast \text{own}(y)(x) \)

Type \( e \):

1. \( \text{Homer} \sim h \)
2. \( \text{Marge} \sim m \)
3. \( \text{Tom} \sim t \)
4. \( \text{Dick} \sim d \)
5. \( \text{Harry} \sim h \)

Type \( \langle t, \langle t, t \rangle \rangle \):

1. \( \text{and}_S \sim \lambda q \lambda p. \ p \land q \)
2. \( \text{or}_S \sim \lambda q \lambda p. \ p \lor q \)

Type \( \langle \langle e, t \rangle, \langle e, t \rangle \rangle \):

1. \( \text{is} \sim \lambda P. P \)
2. \( \text{a} \sim \lambda P. P \)

Type \( \langle \langle e, t \rangle, \langle \langle e, t \rangle, \langle e, t \rangle \rangle \rangle \):
1. \( \text{and}_{\text{VP}} \sim \lambda P' \lambda P \lambda x. P(x) \wedge P'(x) \)

2. \( \text{or}_{\text{VP}} \sim \lambda P' \lambda P \lambda x. P(x) \vee P'(x) \)

Type \( \langle \langle e, \langle e, t \rangle \rangle, \langle \langle e, \langle e, t \rangle \rangle, \langle e, \langle e, t \rangle \rangle \rangle \rangle \):

1. \( \text{and}_V \sim \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \wedge R'(y)(x) \)

2. \( \text{or}_V \sim \lambda R' \lambda R \lambda y \lambda x. R(y)(x) \vee R'(y)(x) \)

Type \( \langle \langle e, t \rangle, e \rangle \):

1. \( \text{the} \sim \lambda P \pi z[P(z) \wedge \forall x[P(x) \rightarrow x \leq z]] \)

Type \( \langle \langle e, t \rangle, \langle e, t \rangle \rangle \):

1. \( -s \sim \lambda P . * P \)

Type \( \langle \langle e, t \rangle, \langle (e, t), t \rangle \rangle \):

1. \( \emptyset_D \sim \lambda P \lambda Q. \exists x[P(x) \wedge Q(x)] \)
11 | Event semantics

11.1 Why event semantics

One of the advantages of translating natural language into logic is that it helps us account for certain entailment relations between natural language sentences. Suppose that whenever a sentence $A$ is true, a sentence $B$ is also true. If the translation of $A$ logically entails that of $B$, then we have an explanation for this entailment. Take the following sentences:

(1) a. Homer smokes and Marge drinks.
   b. ⇒ 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. ⇒ Jones buttered the toast.

Here is how we would represent (3a) given the previous chap-
ters (we are treating the toast as if it was a constant rather than a definite description, but nothing will hinge on this):

(4) \[ \text{Butter}(j,t) \]

If this representation is correct, (3a) is about only two entities: Jones and the toast. Which entity does slowly describe in (3a)? Is it perhaps Jones who is slow? Then we might represent the meaning of that sentence as follows:

(5) \[ \text{Butter}(j,t) \land \text{Slow}(j) \]

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.

(6) a. Jones buttered the bagel quickly.
   b. \[ \text{butter}(J,b) \land \text{Quick}(j) \]

But then, in any model where (5) and (6b) are both true, the following will also be true as a matter of logical consequence!

(7) \[ \text{Slow}(j) \land \text{Quick}(j) \]

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 (Parsons 1990):
(8)  a.  Jones’ buttering of the toast was artful.
    b.  It happened slowly.

So let us assume that in (3a) it is the event of buttering the toast that is slow, and in (6a) it is the event of buttering the bagel that is quick, rather than Jones himself. The two sentences, then, are not only talking about Jones and the things he is buttering but also about the buttering events. According to Davidson (1967), the correct logical representations for (3a) and (6a) are not (5) and (6b) but rather something like the following, where $e$ ranges over events:

(9)  a.  $\exists e. \text{Butter}(j, t, e) \land \text{Slow}(e)$
    b.  $\exists e. \text{butter}(J, b, e) \land \text{Quick}(e)$

A sentence like (3b) 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. ⇒ Jones buttered the toast slowly in the kitchen.
c. ⇒ Jones buttered the toast slowly.
d. ⇒ Jones buttered the toast.

Event semantics provides a straightforward account of these entailment patterns:

(12) a. \( \exists e. \text{Butter}(j, t, e) \land \text{Slow}(e) \land \text{Loc}(e, k) \land \text{Time}(e, \text{noon}) \)
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. John saw Mary leave.
b. ⇒ 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. Brutus stabbed Caesar violently.
b. ⇒ There was something violent.

(16) a. \( \exists e. \text{Stab}(b, c, e) \land \text{Violent}(e) \)
b. \( \exists e. \text{Violent}(e) \)
11.1.1 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:

\[ \exists e. \text{Butter}(e) \land \text{Agent}(e, j) \land \text{Theme}(e, t) \]

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) \[ \exists e. \text{Butter}(e) \land \text{Agent}(e, j) \land \text{Theme}(e, t) \land \text{With}(e, k) \]

One of the advantages of the Neo-Davidsonian view is that it allows us to capture semantic entailment relations between different syntactic subcategorization frames of the same verb, such as causatives and their intransitive counterparts [Parsons 1990]:

(19) a. Mary felled the tree.
b. \[ \Rightarrow \] 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. \[ \Rightarrow \] 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. Jane gave the ball to Mary.
b. Jane 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).

(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 (Krifka 1992).

11.2 Composition in Neo-Davidsonian event semantics

Building Neo-Davidsonian semantics into our fragment requires us to decide how events, event quantifiers, and thematic roles, enter the compositional process. There is currently no universally accepted way to settle the question. A common approach is that verbs and verbal projections (such as VPs and IPs) denote predicates of events and are intersected with their arguments and adjuncts, until an existential quantifier is inserted at the end and binds the event variable (Carlson 1984; Parsons 1990, 1995). A more recent approach views this existential quantifier as part of the lexical entry of the verb, and arguments and adjuncts as adding successive restrictions to this quantifier (Champollion 2015). Both strategies are compatible with the idea that adjuncts and prepositional phrases are essentially conjuncts that apply to the same event. We discuss both of them here. The first approach is more widespread and is sufficient for simple purposes, while the second leads to a cleaner interaction with certain other components of the grammar such as conjunction, negation and quantifiers. There are also other strategies that we will not discuss. For example, Landman (1996) assumes that the lexical entry of a verb consists of an event predicate conjoined with one or more the-
matic roles. [Kratzer (2000)] argues that verbs denote relations between events and their internal arguments while external arguments (subjects) are related to verbs indirectly by theta roles.

### 11.2.1 Verbs as predicates of events

On the first strategy, verbs denote predicates of events:

\[
\begin{align*}
\text{(25) a. } & \quad \text{bark} \rightarrow \lambda e. \text{Bark}(e) \\
\text{b. } & \quad \text{butter} \rightarrow \lambda e. \text{Butter}(e) \\
\text{c. } & \quad \ldots
\end{align*}
\]

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 $\theta$.

\[
\text{(26) Syntax}
\]

\[
\text{DP} \rightarrow \theta \text{ DP}
\]

We then write lexical entries that map these heads suitable roles:

\[
\text{(27) Lexicon}
\]

\[
\theta: \text{ [agent], [theme], } \ldots
\]

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.  \([agent] \sim \lambda \lambda e. \text{agent}(e) = x\)  
  b.  \([theme] \sim \lambda \lambda e. \text{theme}(e) = x\)  
  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 \(\langle v, t \rangle\) is the type of an event predicate.

**Type-Shifting Rule 5. Existential closure**

If \(\alpha \sim \alpha'\), where \(\alpha'\) is of category \(\langle v, t \rangle\), then:

\[
\alpha \sim \exists e. \alpha'(e)
\]

as well (as long as \(e\) does not occur in \(\alpha'\); 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

\[
\begin{align*}
S & \rightarrow \text{AdvP } S \\
\text{VP} & \rightarrow \text{AdvP } \text{VP} \\
\text{VP} & \rightarrow \text{VP } \text{AdvP} \\
\text{AdvP} & \rightarrow \text{Adv}
\end{align*}
\]

(31) Lexicon
Adv: slowly

As we have seen above, *slowly* is interpreted as an event predicate. Its lexical entry is therefore very simple:

(32)  a.  \( slowly \sim \lambda e. \text{Slow}(e) \)

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.

### 11.2.2 A formal fragment

Let us recapitulate the additions to our fragment. The syntax is defined as in three-valued type logic (\( L_\lambda \) with three truth values as in Chapter 8), plus the following additions:

**Syntax rules.** We add the following rule:

(34) **Syntax**

\[
\text{DP} \rightarrow \theta \text{ DP}
\]

**Lexicon.** Lexical items are associated with syntactic categories as follows:

\[
\theta: \quad \text{[agent], [theme], ...}
\]

**Translations**. Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- \( e \) is \( v_0, v \)
- bark, and butter are constants of type \( \langle v, t \rangle \),
- agent and theme are constants of type \( \langle v, e \rangle \).

Type \( \langle v, t \rangle \):
1. \textit{bark} \sim \lambda e. \text{Bark}(e)

2. \textit{butter} \sim \lambda e. \text{Butter}(e)

Type \((v, e)\):

1. \([\text{agent}] \sim \lambda x \lambda e. \text{agent}(e) = x\)

2. \([\text{theme}] \sim \lambda x \lambda e. \text{theme}(e) = x\)

### 11.3 Quantification in event semantics

The system we have seen so far is sufficient for many purposes, including the sentences discussed at the beginning of the chapter. Most papers that use event semantics assume some version of it, although the details differ. Things become more complicated, though, when we bring in quantifiers like \textit{every cat} and \textit{no dog}. As we have seen in Chapter ??, 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 \textit{below} anything else in the sentence. Sentence (35) for example, is not ambiguous. Its only reading corresponds to (36b), where the event quantifier takes low scope). As for (37b), that is not a possible reading of the sentence.

(35) No dog barks.

(36) 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”

(37) 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 (37b) is not a possible reading of sentence (35)?

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 (36b), while applying Hendriks’ object raising rule (or rather, the general schema) to the theta role head leads to the unavailable reading (37b). This is shown in (38) and (39) respectively.
(38) \[ S \]
\[ t \]
\[ \neg \exists x. \text{Dog}(x) \land \exists e. \text{Bark}(e) \land \text{agent}(e) = x \]
\[ \text{DP} \]
\[ \langle (e, t), t \rangle \]
\[ \lambda P \neg \exists x. \text{Dog}(x) \land P(x) \]
\[ \lambda P \]
\[ \langle e, t \rangle \]
\[ \lambda v_1 \exists e. \text{Bark}(e) \land \text{agent}(e) = v_1 \]
\[ 1 \]
\[ S \]
\[ t \]
\[ \exists e. \text{Bark}(e) \land \text{agent}(e) = v_1 \]
\[ \uparrow \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{Bark}(e) \land \text{agent}(e) = v_1 \]
\[ \text{DP} \]
\[ \langle v, t \rangle \]
\[ \lambda e. \text{agent}(e) = v_1 \]
\[ \lambda e. \text{Bark}(e) \]
\[ \text{barks} \]
\[ \theta \]
\[ \langle e, (v, t) \rangle \]
\[ \lambda x \lambda e. \text{agent}(e) = x \]
\[ v_1 \]
\[ \text{[agent]} \]
\[ t_1 \]
The interim conclusion, then, is that event semantics seems to commit us to a quantifier-raising based treatment of quantificational noun phrases.
11.3.1 Verbs as event quantifiers

In the tree in (38), 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:

\[(40)\]

a. \(bark \sim \lambda f \exists e . \text{Bark}(e) \land f(e)\)

b. \(butter \sim \lambda f \exists e . \text{Butter}(e) \land f(e)\)

c. \ldots 

Our grammar will continue to map verbal projections (verbs, VPs and Ss) to the same type. But this type is no longer \(\langle v, t \rangle\) but \(\langle \langle v, t \rangle, t \rangle\). For this reason, we will no longer rely on predicate modification, but instead use function application to combine syntactic arguments with verbal projections. This means that our thematic look more complicated than before:

\[(41)\]

a. \([\text{agent}] \sim \lambda x \lambda V \lambda f . V(\lambda e . \text{agent}(e) = x \land f(e))\)

b. \([\text{theme}] \sim \lambda x \lambda V \lambda f . V(\lambda e . \text{theme}(e) = x \land f(e))\)

c. \ldots 

If the root of the tree is of type \(\langle \langle v, t \rangle, t \rangle\), we need to map it to a truth value. In a simple case such as *Spot barks*, the root will be true of any set of events \(f\) so long as \(f\) contains (possibly among other things) an event that satisfies the relevant event predicate. Whether this is true can be checked by testing whether the set of all events whatsoever, \(\lambda e . \text{true}\), contains such an event:
To formalize this idea, we introduce the type-shifting rule of Quantifier Closure:

**Type-Shifting Rule 6. Quantifier Closure**

If \( \alpha \rightarrow \alpha' \), where \( \alpha' \) is of category \( \langle \langle v, t \rangle, t \rangle \), then:

\[
\alpha \rightarrow \alpha'(\lambda e. \text{true})
\]

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. \(\text{slowly} \sim \lambda V \lambda f . V(\lambda e . \text{Slow}(e) \land f(e))\)

b. \ldots

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.
11.3.2 Another formal fragment

Let us recapitulate the additions to our fragment. The syntax is defined as in three-valued type logic (L_λ with three truth values as in Chapter 8), plus the following additions:

**Syntax rules.** We add the following rule:

\[
(47) \quad \text{Syntax} \quad \quad \text{DP} \rightarrow \theta \text{ DP}
\]

**Lexicon.** Lexical items are associated with syntactic categories as follows:

\[
\theta: \quad \text{[agent], [theme], \ldots}
\]

**Translations.** Verbs get new translations, and we add thematic roles. We will use the following abbreviations:

- \(e\) is \(v_0, v\),
- \(f\) is a variable of type \(\langle v, t \rangle\)
- \(V\) is a variable of type \(\langle \langle v, t \rangle, t \rangle\)
- Bark and Butter are constants of type \(\langle v, t \rangle\)
- agent and theme are constants of type \(\langle v, e \rangle\)

The following entries replace the previous ones:

Type \(\langle v, t \rangle\):

1. \(bark \sim \lambda f \exists e. \text{Bark}(e) \wedge f(e)\)
2. \(butter \sim \lambda f \exists e. \text{Butter}(e) \wedge f(e)\)

Type \(\langle v, e \rangle\):
1. \[ \text{[agent]} \sim \lambda x \lambda \alpha. V(e. \alpha) = x \land f(e) \]

2. \[ \text{[theme]} \sim \lambda x \lambda \beta. V(e. \beta) = x \land f(e) \]

Type \( \langle\langle v, t, t\rangle, \langle\langle v, t, t\rangle, \langle v, t, t\rangle\rangle\rangle \):

1. \[ \text{and}_{\text{VP}} \sim \lambda V' \lambda V' f. V(f) \land V'(f) \]

We have introduced the following type-shifter:

\begin{center}
\textbf{Type-Shifting Rule 7. Quantifier Closure}
\end{center}

If \( \alpha \sim \alpha' \), where \( \alpha' \) is of type \( \langle\langle v, t, t\rangle, \langle\langle v, t, t\rangle, \langle v, t, t\rangle\rangle\rangle \), then:

\[ \alpha \sim \alpha'(\lambda e. \text{true}) \]

as well.

### 11.4 Conjunction in event semantics

In Chapter 10 we have seen that many uses of \textit{and} can be subsumed under a general schema, discussed by Partee & Rooth (1983) among others. This schema is repeated here:

\[ \text{(48)} \quad \text{and}_{\langle\langle \tau, \{\tau, \tau\}\rangle}\rangle \]

\[ \sim \begin{cases} 
\lambda q \lambda p. p \land q & \text{if } \tau = t \\
\lambda X_T \lambda Y_T \lambda Z_{\sigma_1} \cdot \langle\langle \text{and}\rangle\rangle_{\langle\sigma_2, \{\sigma_2, \sigma_2\}\rangle}(X(Z))(Y(Z)) & \text{if } \tau = \langle\sigma_1, \sigma_2\rangle 
\end{cases} \]

where \( \langle\langle \text{and}\rangle\rangle_{\langle\sigma_2, \{\sigma_2, \sigma_2\}\rangle} \) denotes the translation of \textit{and} for the corresponding type.

What does this rule amount to in the case of VP-modifying \textit{and}, as in \textit{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 (48) in each case results in the following:
Exercise 2. Show how rule (48) leads to these two representations.

As you can see in (51) and (52), these two choices lead to very different translations: (50a) and (50b) respectively.

Exercise 3. Add slowly and quickly to the tree in (52) and show how the resulting formula avoids the attribution of contradictory properties to the same event.
(51) \[
S_t \\
\exists e. \text{smoke}(e) \land \text{drink}(e) \\
\land \text{agent}(e) = h \\
\uparrow \\
\langle v, t \rangle \\
\lambda e. \text{smoke}(e) \land \text{drink}(e) \\
\land \text{agent}(e) = h
\]
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 en-
try for VP-level conjunction that is compatible with event predicates. This is similar to DP-level conjunction, where in Chapter 10 we have encountered both schema-based and non-schema-based entries.

**Exercise 4.** Formulate an entry for VP-level conjunction that is compatible with the event-predicate based approach. Hint: use sums of events. Make sure it predicts the right truth conditions for *Homer smoked slowly and drank quickly*. Assume that for any theta role $\theta$, $\theta(e \oplus e') = \theta(e) \oplus \theta(e')$. 

11.5 Negation in event semantics

(53)

\[ S_t \]
\[ \exists e. \neg \text{bark}(e) \land \text{agent}(e) = s \]
\[ \uparrow \]
\[ \langle v, t \rangle \]
\[ \lambda e. \neg \text{bark}(e) \land \text{agent}(e) = s \]

[Diagram of the syntactic tree for the sentence showing negation]
(54) $S$

\[
\begin{align*}
&\neg \exists e. \text{bark}(e) \land \\
&\text{agent}(e) = s
\end{align*}
\]

$\uparrow$

\[
\begin{align*}
\langle \langle v, t \rangle, t \rangle
\end{align*}
\]

$\lambda f \neg \exists e. \text{bark}(e) \land$

\[
\begin{align*}
\text{agent}(e) = s \land f(e)
\end{align*}
\]
12 Tense

12.1 Introduction

So far, we have been ignoring the contribution of tense. In this chapter, we will finally face it. In order to do so, we must grapple with the related issue of aspect. Ideally, our theory should be able to explain the contrasts in meaning among all of the following forms.

(1) a. Ann dances.  [simple present]
b. Ann danced.    [simple past]
c. Ann will dance. [simple future]

(2) a. Ann is dancing.  [present progressive]
b. Ann was dancing. [past progressive]
c. Ann will be dancing. [future progressive]

(3) a. Ann has danced. [present perfect]
b. Ann had danced. [past perfect]
c. Ann will have danced. [future perfect]

(4) a. Ann has been dancing. [present perfect progressive]
b. Ann had been dancing. [past perfect progressive]
c. Ann will have been dancing. [future perfect prog.]

We begin with aspect, in both of its senses, and then move on to tense.
12.2 Aspect

12.2.1 Aktionsart

The term aspect can refer to two different things in linguistic theory. Both have to do with the temporal properties of a state or event being described or referred to. We begin with aktionsart, a German word that literally means ‘type of action’. In a famous paper entitled Verbs and Times, Vendler (1967) distinguished between four types of eventualities:

- States (example: have a tan)
- Activities (example: make sandcastles)
- Accomplishments (example: run a mile)
- Achievements (example reach the pier)

A fifth type, namely ‘semelfactives’, was later added (example: jump). These are categories of states or events—eventualities, to be neutral between state and event—with various different properties.

One dimension along which these different eventuality types differ is telicity. A telic eventuality has an endpoint; telos means ‘goal’ in Greek. Verb phrases denoting telic eventuality types can be modified with in-adverbials such as in an hour. Compare:

(5) a. Ida ran a mile in an hour. [accomplishment]
   b. ??Ida made sandcastles in an hour. [activity]

Read a poem is telic, while read poetry is telic.

Verb phrases denoting atelic eventualities, on the other hand, are more natural in combination with for-adverbials such as for an hour:

\footnote{Other words for ‘aktionsart’ include lexical aspect, situation aspect, internal aspect, aspectual class, and situation type.}
(6)  
   a. ??Ida ran a mile for an hour.  [accomplishment]  
   b. Ida made sandcastles for an hour.  [activity]

States and activities are atelic, while accomplishments, achievements, and semelfactives are telic.

What distinguishes states from activities is that activities are **DYNAMIC** while states are not. For example, when one reads poetry, physical and psychological changes take place. The state/non-state distinction also has reflexes in the grammar, as the progressive in English does not combine well with stative predicates:

(7)  
   a. Ida is running along the beach.  [activity]  
   b. ??Ida is having a tan.  [state]

Furthermore, the simple present tense gives rise to a habitual interpretation with non-states:

(8)  
   a. Ida runs along the beach.  [activity: habitual]  
   b. Ida has a tan.  [state: non-habitual]

   What distinguishes accomplishments from achievements and semelfactives is that the former are **DURATIVE** while the latter are conceptualized as taking place at a single moment. This contrast can be observed in conjunction with *in* phrases. To see this, consider the following sentences:

(9)  
   a. Ida will run a mile in 20 minutes.  [accomplishment: in = duration]  
   b. Ida will reach the pier in 20 minutes.  [achievement: in = after]  
   c. Ida will jump in 20 minutes.  [semelfactive: in = after]

With the accomplishment *run a mile*, 20 minutes describes the duration of the running-a-mile event, while with the achievement *reach the pier* and the semelfactive *jump*, 20 minutes is the time that will elapse before the event takes place.

Finally, what distinguishes achievements from semelfactives
is that the former involve a change of state while the latter do not. Because semelfactives do not involve a change of state, they can be iterated, and an iterative reading arises with for adverbials:

(10)  a. Ida jumped for an hour. [semelfactive: iterative]
    b. ??Ida reached the pier for an hour. [achievement]

Repeated jumping is an eventuality type that is atelic, unlike jumping once, which is telic. The repetition induced by the for adverbial here can be seen as a secondary operation on the meaning of the verb *jump*, taking it from its basic telic meaning to an atelic meaning involving iteration of the basic meaning.

The kind of eventuality being described can depend on the object of the verb. For example, *make sandcastles* is atelic, while *make a sandcastle* is telic. Thus it is not verbs but verb phrases that are appropriate to classify with respect to their aktionsart. But as we have just seen in the case of semelfactives, there may be other elements in a sentence that help to determine the aspectual properties of the eventuality being described by the entire sentence.

12.2.2 Viewpoint aspect

We turn now to the other kind of aspect, which goes by many names, including viewpoint aspect, grammatical aspect (vs. lexical aspect), and perspective point. We choose the term VIEWPOINT ASPECT here, emphasizing the idea that it has to do with how an eventuality is ‘viewed’, not with its inherent temporal properties. According to a prominent view on viewpoint aspect (Klein [1994]), this kind of aspect provides a link between eventualities and tense, by specifying the relation between the EVENT TIME and a REFERENCE TIME, two concepts which we will explain shortly.

English has two morphological forms that express viewpoint aspect: the perfect, as in *I have eaten*, and progressive, as in *I am eating*. Confusingly enough, the English progressive expresses IM-
PERFECTIVE ASPECT. Stative predicates like *have a tan* are also imperfective. So the two main aspectual distinctions that English is sensitive to are perfect vs. non-perfect, and perfective vs. imperfective. Confusingly enough, ‘perfective’ is totally different from ‘perfect’; these categories can cross-classify:

<table>
<thead>
<tr>
<th></th>
<th>PERFECTIVE</th>
<th>IMPERFECTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERFECT</td>
<td><em>I have danced</em></td>
<td><em>I have been dancing</em></td>
</tr>
<tr>
<td>NON-PERFECT</td>
<td><em>I danced</em></td>
<td><em>I was dancing</em></td>
</tr>
</tbody>
</table>

That said, it is thought that there is a historical trajectory from perfect to perfective, starting with resultative:

RESULTATIVE >> PERFECT >> PERFECTIVE

So there is a connection, albeit historical.

The English perfect is a key motivation for Reichenbach’s (1947) theory of temporal reference. Reichenbach noticed that in order to give a good theory of the English perfect, it is necessary to consider not only the time of utterance and the time at which the event occurred, but also a more abstract time that he referred to as **REFERENCE TIME** (later called **TOPIC TIME**).

- **SPEECH TIME** (*S*): the time the sentence is uttered
- **EVENT TIME** (*E*): the time the event takes place
- **REFERENCE TIME** (*R*): the time under discussion (also known as **Topic Time**)

The concept of ‘reference time’ was Reichenbach’s major innovation, and the concept that is least intuitively obvious. One way of characterizing it is that it is the time that the sentence is ‘about’ (hence the term ‘topic time’).

---

2 See Condoravdi and Deo, ‘Aspect shifts in Indo-Aryan and trajectories of semantic change’.

3 This section borrows quite liberally from Cable’s 2008 notes on tense.
According to Reichenbach, the difference between a sentence like *I had danced* and a sentence like *I danced* is that in the former case, with the perfect, the sentence is about a time prior to speech time before which some dancing took place (so $E < R < S$):

$$E \quad R \quad S$$

In the case of the simple past, *I danced*, the sentence is about the time at which the dancing took place (so $E = R < S$), a time prior to speech time.

$$E, R \quad S$$

Aside from its intuitive appeal, support for this analysis comes from the fact that temporal adverbs like *at 5pm* track the hypothesized reference time:

(11)  
   a. At 5pm, I danced. (5pm = dancing time)
   b. At 5pm, I had danced. (dancing time < 5pm)

Assuming the modifier is identified with the reference time, we correctly predict that 5pm is the time of dancing for the sentence in simple past, but a time before the speech time and after the time of dancing in the past perfect.

The contribution of the perfect, then, is that the event time precedes the reference time. This holds in other tenses as well. Consider the present tense:

(12)  
   a. I have danced.
   b. I dance.

The simple present example in (12b) has a habitual present interpretation; the present perfect example expresses that a dancing event took place prior to speech time. This can be understood if
the present tense puts reference time at speech time, and the contribution of the perfect is to locate the event time as being prior to reference time. The present perfect thus looks like this:

\[ E \rightarrow R, S \]

And the simple present looks like this:

\[ E, R, S \]

This theory of the perfect works in the future tense as well. If we assume that the future tense locates the reference time after speech time, and the perfect locates the event time prior to the reference time, a sentence like:

(13) At 5pm, I will have danced.

is predicted to imply that the time referred to by 5pm is in the future, and that some dancing event will have occurred prior to that (perhaps before, perhaps after speech time). This accords with intuition. In contrast:

(14) At 5pm, I will dance.

implies that the time referred to by 5pm is in the future, and that a dancing event will take place at that time.

**Exercise 1.** With the tools just developed, explain the following contrast.

(15) How unfortunate!

a. Now that John tells me this I have mailed the letter.

b. #Now that John tells me this I mailed the letter.
The picture we arrive at, then, is that the contribution of tense is to relate the reference time with the speech time, and the contribution of viewpoint aspect is the relate the event time to the reference time. Past tense locates the reference time prior to speech time; present tense sets them equal; and future locates reference time after speech time. The perfect places the event prior to reference time; otherwise the event takes place at reference time.

It would be irresponsible for us not to mention that this view is somewhat of an oversimplification; there are a number of different uses for the English perfect, including:

\begin{enumerate}
\item Ed has put the cake in the oven. \textbf{RESULTATIVE}
\item Ed has visited Korea many times. \textbf{EXISTENTIAL}
\item Ed has lived in Korea for 3 years. \textbf{UNIVERSAL}
\end{enumerate}

These examples are taken from Condoravdi and Deo, ‘Aspect shifts in Indo-Aryan and trajectories of semantic change’. We will however set these uses aside.

Let us now turn to perfective vs. imperfective aspect, which also relates the event time to the reference time. Consider the following contrast:

\begin{enumerate}
\item At 5pm, I danced.
\item At 5pm, I was dancing.
\end{enumerate}

In the first case, there was a dancing event that took place at 5pm. In the second case, there was a dancing event that extended across 5pm. Klein’s (1994) proposes to model this using inclusion among time intervals. In the past progressive example the time interval during which the dancing event took place includes the reference time interval identified with 5pm. This gives us the following picture for past progressive:

\begin{equation}
\begin{array}{c}
R \\
E \\
S
\end{array}
\end{equation}
In general, imperfective aspect signals that the event time contains the reference time, while imperfective aspect signals the reverse: the reference time contains the event time.

This notion of ‘containedness’ entails a view of times on which they are actually stretches of time, or time intervals. A time $t$ contains another time $t'$ if every moment included within $t'$ is also included in $t$. Using $\subseteq$ to represent this containment relationship, we can represent the contribution of aspectual morphology as follows:

- perfective aspect: $E \subseteq R$
- imperfective aspect: $R \subseteq E$

Note that stative predicates also have imperfective aspect:

(18) At 5pm, I was asleep.

As with the progressive, the time of the sleeping eventuality includes the reference time. Thus progressive is a specific type of imperfective.

To summarize, tense relates speech time with reference time, and aspect relates reference time with event time. In the following sections, we will work towards a formal and compositional implementation of these ideas, which will require us to both establish the machinery for talking about indexicality and make certain decisions about exactly how to consider reference time: Is it more like an existentially bound variable or more like a free variable? We turn first the issue of indexicality.

### 12.3 Indexicality

The notion of ‘speech time’ is an indexical one: It has to do with the so-called context of utterance, or context of use. In Chapter 9 on dynamic semantics, we spoke of context in terms of the information that has been established so far in the discourse,
and the discourse referents that had been introduced, and how meanings could be seen as operations that update such contexts. The ‘context of utterance’ is context in another sense: It’s about how is speaking, to whom, where, when, etc.

An **INDEXICAL** may be defined as “a word whose referent is dependent on the context of use, which provides a rule which determines the referent in terms of certain aspects of the context” (Kaplan, 1977, 490). Examples include *I, my, you, that, this, here, now, tomorrow, yesterday, actual*, and *present*. Kaplan distinguishes between two sorts of indexicals:

- **DEMONSTRATIVES**: indexicals that require an associated demonstration. Examples: *this* and *that*.

- **PURE INDEXICALS**: indexicals for which no demonstration is required. Examples: *I, now, here, tomorrow* (although *here* has a demonstrative use: “In two weeks, I will be here [pointing]”)

To warm up intuitions regarding how indexicals should be analyzed, consider the following two utterances:

(19) a. (May 11, 2010, uttered by Elizabeth Coppock:) I am turning 30 today.  
    b. (May 12, 2010, uttered by Elizabeth Coppock:) I am turning 30 today.

What do you think: Do they have the same meaning or different meaning? How about the following pair:

(20) a. (May 11, 2010, uttered by Elizabeth Coppock:) I am turning 30 today.  
    b. (May 12, 2010, uttered by Elizabeth Coppock:) I turned 30 yesterday.

It’s not immediately obvious how to answer this question. In some sense, the examples in [(19b)] have the same meaning, but in another sense, those in [(20b)] have the same meaning.
Kaplan resolves this tension by distinguishing between two levels of meaning, CHARACTER and CONTENT.

- CHARACTER is the aspect of meaning that two utterances of the same sentence share across different contexts of utterance.

- 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 (19b) have the same character, while the pair in (20b) have the same content. Kaplan’s ‘contents’ are essentially the same as Carnap’s ‘intensions’; they are functions from possible worlds (a.k.a. ‘circumstances of evaluation’) to extensions. For sentences, the content is a proposition, a function from possible worlds to truth values. The character of a sentence is something that, given a context of utterance, gives you a content; formally a function from contexts to contents. So in a nutshell, the Kaplanian picture is as follows:

- Character + Context of utterance ⇒ Content

- Content + Circumstance of evaluation ⇒ Extension

(We could have written ‘Intension’ in place of ‘Content’ here; they play indistinguishable roles for our purposes.)

The CONTEXT OF UTTERANCE determines who is speaking, to whom, when, where, and in what possible world.

\[ c = (sp, ad, t, loc, w) \]

In Kaplan’s ‘logic of indexicals’, there are certain special indexical constants, whose semantics are defined as follows:

\begin{align*}
(21) \quad & a. \quad [i]_{M,g,c} = sp(c) \\
& b. \quad [u]_{M,g,c} = ad(c) \\
& c. \quad [\text{now}]_{M,g,c} = i(c) \\
& d. \quad [\text{here}]_{M,g,c} = loc(c)
\end{align*}
English expressions can then be mapped to these special indexical constants like so:

(22)  
   a.  $I \sim i$
   b.  $you \sim u$
   c.  $now \sim n$
   d.  $here \sim here$

Again, the content of a sentence is the proposition that is expressed after the reference of all of the indexicals has been fixed by the context of utterance. Formally, fixing $g$ and $c$, the content of $\phi$ can be defined as:

$$\{ M : \sem{\phi}^{M,g,c} = 1 \}$$

And again, the character of a sentence is that aspect of its meaning that is the same across different contexts of use. This notion can be formalized as a function from contexts of utterance to contents. Fixing $g$, the character is that $f$ such that:

$$f(c) = \{ M : \sem{\phi}^{M,g,c} = 1 \}$$

Now, Kaplan actually argues for this view of indexicals against an alternative theory according to which indexicals are disguised definite descriptions. One might imagine the following alternative analysis:

(23)  $I \sim ix.\text{Speaker}(x)$
(24)  $you \sim ix.\text{Addressee}(x)$

On this view, here is no need to posit a separate context of utterance.

---

The way Kaplan (1989) really defines it is closer to:

$$\{ w : \sem{\phi}^{M,g,c,w} = 1 \}$$

where $M = \langle D, I, W, C \rangle$ is an intensional model. We will introduce intensional models in Chapter 13.
This alternative view fails to account for the fact that the following two sentences have very different meanings.

(25) a. If I were male, I would not be speaking right now.
    b. If the person speaking were male, I would not be speaking right now.

Spoken by a female person, the first sentence would seem untrue. But the second sentence would seem true. The situation with proper names is analogous to the case with indexicals:

(26) a. If Liz were male, Liz would not be speaking right now.
    b. If the person speaking were male, Liz would not be speaking right now.

The first sentence seems false, but the second sentence seems true. In any case, their meanings are very different.

Similarly, Ed's wish in the following two sentences is satisfied under very different circumstances:

(27) a. Ed wishes that I were male.
    b. Ed wishes that the person speaking were male.

And again, the variant with the proper name patterns with the one with the indexical first person pronoun:

(28) Ed wishes that Liz were male.

According to Kaplan, indexicals, like proper names, are DIRECTLY REFERENTIAL: they refer to the same individual in every possible world. Unlike definite descriptions like the speaker, they do not look in a world to see who is the speaker there and then refer to that person. They directly pick out an element of the context of utterance. Definite descriptions like the speaker, in contrast, may refer to different individuals in different worlds. Although indexicals may be said to have descriptive content, it is part of their character, not their content.
Kaplan’s conclusion is that we need to add CONTEXT OF UTTERANCE as a parameter according to which we determine the semantic value of linguistic expressions:

\[ [\alpha]^{M,g,c} = \ldots \]

We will adopt this context-of-utterance parameter in our treatment of tense, which as Reichenbach (1947) points out, is a “particularly important” type of indexicality.

### 12.4 Tense

We are now ready to begin formalizing a theory of tense, building on the ideas of Reichenbach, Klein, and Kaplan. One of the questions that we must take a stand on is how to conceive of the reference time: Is it existentially quantified or free? In other words, does the past tense say ‘there was some time \( R \) in the past such that...’ or does it pick out a salient time \( R \) from the context?

In order to give a bit of historical context for early work on tense, let us begin by present Arthur Prior’s tense logic, which is essentially an ‘existential’ theory of the past. We will then discuss its shortcomings, in favor of an ‘anaphoric’ theory of the past.

#### 12.4.1 Priorean tense logic

In Arthur Prior’s TENSE LOGIC, a formula can vary in its truth value across time. Thus Susan is asleep might be true at time \( t \), but false at time \( t' \). A future sentence like Susan will be asleep can then be said to be true at time \( t \) if there is a time \( t' \) later than \( t \) at which Susan is asleep.

To achieve this, we will add to our models so that they consist not only of a domain of individuals \( D \) and an interpretation function \( I \) but also a set of times \( T \) and a linear ordering relation \( < \) among the times. A TEMPORAL MODEL for a language \( L \) is then a
quadruple
\[ \langle D, I, T, < \rangle \]
such that \( D \) is a set of individuals, \( T \) is a set of times, \( < \) is the ‘earlier than’ relation among the times, and \( I \) is an interpretation function which maps the non-logical constants to appropriate denotations at the various times. The function \( I \) will thus take two arguments: a constant, and a time. For example, suppose we have a model in which the domain \( D = \{ a, b, c \} \), the set of times \( T = \{ t_1, t_2, t_3 \} \), and we have two individual constants john and mary, and one predicate constant Happy. The interpretation function \( I \) might then be defined as follows:

\[
\begin{align*}
I(t_1, \text{john}) &= b \\
I(t_1, \text{mary}) &= a \\
I(t_1, \text{Happy}) &= \{ a, b, c \}
\end{align*}
\]

\[
\begin{align*}
I(t_2, \text{john}) &= b \\
I(t_2, \text{mary}) &= a \\
I(t_2, \text{Happy}) &= \{ a, b \}
\end{align*}
\]

\[
\begin{align*}
I(t_3, \text{john}) &= b \\
I(t_3, \text{mary}) &= a \\
I(t_3, \text{Happy}) &= \{ c \}
\end{align*}
\]

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:

\[
\begin{align*}
\llbracket \text{Happy}(\text{mary}) \rrbracket^{M,g,t_1} &= 1 \\
\llbracket \text{Happy}(\text{mary}) \rrbracket^{M,g,t_3} &= 0
\end{align*}
\]

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:

- \([F\phi]_{M,g,t} = 1\) iff \([\phi]_{M,g,t'} = 1\) for some \(t'\) such that \(t < t'\).
- \([P\phi]_{M,g,t} = 1\) iff \([\phi]_{M,g,t'} = 1\) for some \(t'\) such that \(t' < t\).

The way we have set things up, these formulas can be iterated ad infinitum, letting us model statements like ‘Susan will have seen the report’ which can take the form of \(FP\phi\) or ‘A child was born that would become the ruler of the world’ (Kamp, 1971), which might be modeled using a future operator in the scope of a past operator.

But let us not get too married to this system, because it suffers from a number of difficulties as a theory of tense. We turn to these next.

### 12.4.2 Shortcomings of the Priorean theory of tense

#### 12.4.2.1 Partee's example

In Prior’s tense logic, as we have just discussed, there is an operator \(P\) (for ‘past’) whose semantics is defined such that

\[
P\phi
\]

is true at time \(t\) if \(there is some time t' prior to t\) such that \(\phi\) is true at \(t'\). For example, ‘John sneezed’ would be true at \(t\) if there is some time \(t'\) prior to \(t\) such that John sneezed at \(t'\). This amounts to an existential theory of the past tense.

But consider a context in which you’ve just baked some cookies, and are on the way over to your friend’s house. You realize mid-journey that you left the oven on. Then you say:
The existential theory of the past tense does not make correct predictions about this case, as Partee (1973) famously pointed out. We could consider two possible scopes for negation relative to the past tense:

- **Negation scopes over existential past tense** (NOT > PAST): It is not the case that there is a time in the past when I turned off the stove.

- **Existential past tense scopes over negation** (PAST > NOT): There is a time in the past when I didn't turn off the stove.

Neither one of these is right. The first one is too strong – surely there is some time in the past when you turned off the stove. The second one is too weak – of course there is a time in the past when you didn't turn off the stove! For example, consider the moment you put the cookies in the oven; you didn't turn off the stove then. It seems that (29) is saying something about a particular time.

Partee (1973) notes a number of structural parallels between tenses and pronouns, in support of the so-called REFERENTIAL THEORY OF TENSE. On this view, the past tense in a sentence like (29) is similar to a free pronoun, anaphorically referring back to a time that has previously been introduced into the discourse.

### 12.4.2.2 Interactions between tense and aspect

Another shortcoming of Prior’s theory of tense is that it has nothing to say about the interaction between tense and aspect.

---

5Here we are not primarily concerned with the various aspectual classes most famously laid out by Vendler (1957) (state, activity, accomplishment, achievement, and other distinctions of this kind). These types of distinctions fall under the heading of AKTIONSART (type of event), also known as ‘situation aspect’, ‘lexical aspect’, or ‘inner aspect’. The distinction between perfective and imperfective falls under the heading of ‘viewpoint aspect’, also known as ‘grammatical aspect’, or ‘outer aspect’. Viewpoint aspect can be thought of as locating events...
example, both of the following sentences are in the past tense, but one implies that the event is complete, and the other allows for the possibility that the event is continuing:

(30) (When I was in the room,) Dave ate the cookie. (perfective)
(31) (When I was in the room,) Dave was eating the cookie. (imperfective)

The example in (30), which is in the simple past, has **perfective aspect**. With perfective aspect, the past tense implies that the event in question has been completed. The past progressive example in (31) has **imperfective aspect**, which does not have the same implication; the event might still be going on. As pointed out by Klein (1994), this shows that it is not always the case that a past tense sentence means that the event described is in the past.

12.5 **A formal theory of tense**

12.5.1 **Anaphoric theory of the past**

Let us now present a theory of the past on which it refers to a salient past time, as Partee advocates. We will incorporate the ideas of Reichenbach, Klein and Kaplan in our theory as well.

As in Priorian tense logic, a model for our formal language specifies a set of times \( T \), along with an ordering relation among the times \( < \) as well as a containment relation among the times \( \subseteq \). We extend the models for intensional logic that we had before, so a model \( M \) will have the following structure:

\[
M = \{ D, I, W, T, <, \subseteq \}
\]

with respect to a point of view in the sense that the perfective “looks at the event from the outside”, and the progressive “looks at the event from the inside” (Comrie 1976; Smith 1997); see Bhatt & Pancheva (2005) for a pedagogical discussion of aspect.
where

- $D$ is the domain of individuals $D$
- $I$ is an interpretation function assigning semantic values to each of the non-logical constants in the language
- $W$ is a set of worlds
- $T$ is a set of times
- $<$ is a precedence relation among times
- $\subseteq$ is a containment relation among times

What constitutes the utterance time depends of course on the context of utterance, which means that tense morphology is indexical. Therefore, to model tense we will use an extension of Kaplan’s system, where the semantic value of an expression is determined relative to a model $M$, an assignment function $g$, a world $w$, and a context $c$.

$$[[\alpha]]^{M,g,w,c}$$

The ‘utterance time’ is the time determined by $c$, which we call $t(c)$.

Our formal language will allow expressions that refer to times. We will use $i$ as the type designator for times, so expressions that refer to times will be of type $i$. We will allow an infinite set of variables of type $i$, so for example

$$v_{3,i}$$

is a variable of type $i$ with index 3. We use

$$t_n$$

as an abbreviation for

$$v_{n,i}.$$
• Syntax: If $\alpha$ and $\beta$ are expressions of type $i$, then,

$$\alpha < \beta$$

is a formula.

• Semantics:

$$[\alpha < \beta]_{M,g,w,c} = \begin{cases} 1 & \text{if } [\alpha]_{M,g,w,c} < [\beta]_{M,g,w,c} \\ 0 & \text{otherwise} \end{cases}$$

(where $<$ is determined by $M$)

Similarly:

• Syntax: If $\alpha$ and $\beta$ are expressions of type $i$, then, $\alpha \subseteq \beta$ is a formula.

• Semantics:

$$[\alpha \subseteq \beta]_{M,g,w,c} = \begin{cases} 1 & \text{if } [\alpha]_{M,g,w,c} \subseteq [\beta]_{M,g,w,c} \\ 0 & \text{otherwise} \end{cases}$$

(where $\subseteq$ is determined by $M$)

The expression $t \subseteq t'$ can be read, ‘$t$ is contained in $t’$. Thus $t'$ is the (potentially) larger interval, occupying a stretch of time that contains the stretch of time $t$ occupies.

With these tools in hand, let us outline a simple theory of tense. The basic idea is that the past tense denotes a variable over times. We assume that the natural language morpheme $\text{PAST}$ is associated with an index $n$, just like a pronoun. This index determines the variable over times that the past tense morpheme maps to.

(32) \[ \text{PAST}_n \rightsquigarrow t_n \] (first version)

But there is an additional constraint. The past tense further requires that $t_n$ precedes the time of utterance, while the present tense requires that $t_n$ is identical to the time of utterance. We use the constant $\text{now}$ to denote the time of utterance:
As Heim (1994) discusses, the denotation of the past tense should be undefined unless \( t_n < \text{now} \) holds, because this constraint is more like a presupposition than an entailment. (If it were otherwise, then it should be possible to target the constraint with negation, and \( I \text{ didn't turn off the stove} \) could be true in virtue of there being a non-past time at which the speaker turns off the stove.) As long as that constraint holds, the past tense should be the value of the assignment function for \( t_n \). To get this result using the formal tools at our disposal, we can use an \( \iota \)-expression, as follows:

\[
(34) \quad \text{PAST}_n \sim \iota t \cdot [ t = t_n \wedge t_n < \text{now} ]
\]

This expression will constrain both the assignment function \( g \) and the context of utterance \( c \). The past tense will only have a defined value relative to assignment function \( g \) and context of utterance \( c \) when \( g(t_n) \) precedes the time of utterance \( t(c) \).

For now, the present tense will be analyzed simply as:

\[
(35) \quad \text{PRESENT} \sim \text{now}
\]

although there is evidence that the present tense behaves somewhat differently from the word \text{now} (Kamp, 1971). Compare:

\[
(36) \quad \begin{align*}
\text{a. Someday Susan will marry a man she loves.} \\
\text{b. Someday Susan will marry a man she loves now.}
\end{align*}
\]

These two sentences mean something different; the former describes a man she will love in the future; the latter describes a man she loves now. This contrast can be captured using Kratzer’s (1998) notion of ‘zero tense’. A ‘zero tense’ for Kratzer is an indexed time variable with no presuppositions (hence the name ‘zero’),

---

This presupposition is analogous to the presupposition on gender features on pronouns: \( he \) presupposes that the reference is male. That presupposition can be captured by mapping \( he_i \) to the expression \( \iota x . [ x = v_i \wedge \text{male}(v_i) ] \).
which must be bound by a local antecedent. We will maintain
the simple theory of the present tense in (35) for the time being,
though.

Kratzer (1998) proposes that the syntax of verb phrases is lay-
ered so that an aspectual phrase, where the perfective/imperfective
distinction is represented, dominates the VP, and a tense phrase,
where the past/present distinction is represented, in turn domi-
nates the aspectual phrase:

\[
\text{TenseP} \\
\text{Tense} \quad \text{AspP} \\
\text{Asp} \quad \ldots
\]

The node that AspP dominates is taken to denote a property of
times, type \(\langle i, t\rangle\). The AspP node imposes further constraints on
this property of times, and this property is predicated of the time
denoted by the Tense node.

Verbal predicates will take time arguments. For example:

(37) \(\text{dance} \rightsquigarrow \lambda x. \lambda t. \text{Dance}(t, x)\)

This expression will thus denote a function from individuals to
functions from times to truth values. So assuming that \(\text{Ann} \rightsquigarrow a\),
\(\text{Ann dance}\) will be interpreted as:

\(\lambda t. \text{Dance}(t, a)\)

Note that in an event-semantic framework, it is assumed that
verbs like \textit{dance} denote properties of events. These events are
assumed to have ‘temporal traces’ – the interval of time during
which they occur. The temporal trace of an event \(e\) is usually de-
noted \(\tau(e)\). In such a framework, the temporal argument of a verb

Kratzer (1998) analogizes zero tenses to the phenomenon observed in sen-
tences like \textit{Only I did my homework}, where the first person possessive pronoun
\textit{my} seems to be interpreted without its first person feature, because the sentence
can mean ‘I am the only person \(x\) such that \(x\) did \(x\)’s homework’, not ‘I am the
only person \(x\) such that \(x\) did my (the speaker’s) homework.’
would be introduced separately, yielding a predicate of times like:

\[ \lambda t. \exists e [\text{Dancing}(e) \land \text{Agent}(e, a) \land \tau(e) = t] \]

for Ann dance. This could then combine with aspectual and tense morphology in the same way, which we are about to see.

An Asp node will dominate either \textsc{perf} for ‘perfective’ or \textsc{imp} for ‘imperfective’. Perfective aspect has the following interpretation:

\begin{equation}
\text{PERF} \sim \lambda P_{(i,t)} \cdot \lambda t. \exists t'. t' \subseteq t \land P(t')
\end{equation}

‘Takes a predicate of times \( P \), and returns a predicate of times that is true of a time \( t \) if \( t' \) is contained in a time \( t \) at which \( P \) is true.’

\begin{equation}
\text{IMP} \sim \lambda P_{(i,t)} \cdot \lambda t. \exists t'. t \subseteq t' \land P(t')
\end{equation}

‘Takes a predicate of times \( P \), and returns a predicate of times that is true of a time \( t \) if \( t \) is contained in a time \( t' \) at which \( P \) is true.’

The event time in these formulas corresponds to \( t' \), because that is the time of which \( P \) is predicated; cf. \( P(t') \) in the formula. The topic time is \( t \). The difference between perfective and imperfective aspect is captured by the underlined portion in the informal glosses: With perfective aspect, the requirement is that the topic time \( t \) contains the event time \( t' \). With imperfective aspect, the requirement is the other way around: that the topic time \( t \) is contained in the event time \( t' \).

Thus for an example像 Anne danced, we have the derivation in Figure [12.1]. The top node introduces the presupposition \( t_n < \text{now} \). This presupposition can be extracted from the formula, yielding the simpler formula:

\[ \exists t'. t' \subseteq t_n \land \partial(t_n < \text{now}) \land \text{Dance}(t', a) \]

In this formula (and the equivalent one at the top of the tree in Figure [12.1], \( t_n \) is a free variable over times that is presupposed
Figure 12.1: Derivation for *Ann danced*.

to precede the moment of speech. The discourse context should provide an assignment function that will give a value to this free variable. As long as the value is one that precedes the time of utterance, the sentence will have a defined truth value.

**Exercise 2.** Write out how you would read the formula at the top of Figure 12.1 aloud.

**Exercise 3.** Explain how this treatment explains the ‘completion inference’ of the past perfect, i.e., the fact that *Ann danced* implies that there is a dancing event carried out by Ann that has reached completion.

**Exercise 4.** Compute a tree for *Ann was dancing.*
12.6 Future (in English)

It is natural to suppose that the English verb *will* denotes a time located after utterance time. However, many authors claim that the future is not a true tense. Evidence for this idea comes from the fact that there seems to be a past-tense version of *will*, namely *would*, seen in:

(40) (In 1981, Dave’s marriage was very stable.) However, he **would** later learn (in 1987) that his wife was cheating on him.

This sentence means that, spoken in 1981, the sentence “Dave **will** learn that his wife is cheating” is true. If there is a past version of *will*, then *will* must represent the combination of two elements, a tense element and something else.

The other element is thought to be akin to German *woll*, which we represent **WOLL** and treat on a par with the aspectual markers **PERF** and **IMP**.

(41)  **WOLL** $\sim \lambda P_{\langle i, t \rangle} \cdot \lambda t . [\exists t'. t < t' \land P(t')]$

This can combine with both present and past morphology. The verb *will* is the combination of present tense with **WOLL**; the verb *would* combines past and **WOLL**. A sentence like *Anne will dance* will have the representation in Figure [12.2] then.

On this view, English *will* does not occupy the same syntactic position as present or past morphology, nor does it describe the relationship between topic time and utterance time. Rather, it combines **WOLL** which is more like an aspectual marker with one of the two tenses, present or past. Thus the ‘future’ is not a tense in English. A similar claim has been made for S’át’imets (Salish) by [Matthewson (2006)](https://doi.org/10.1017/CBO9780511628195).
Exercise 5. Give a tree for Ann would dance.

12.6.1 Sequence of tense

One phenomenon that we have not covered is so-called ‘sequence of tense’ phenomena. Examples include the following:

(42) John decided a week ago that in ten days he would say to his mother that they were having their last meal together. (Abusch 1988)

(43) John said he would buy a fish that was still alive. (Ogihara 1989)

(44) Mary predicted that she would know that she was pregnant the minute she got pregnant. (Kratzer 1998)

In each of these examples, the bolded phrase is morphologically
past tense, but is not interpreted as such. Several authors, starting with Ogihara (1989), have suggested that the tense feature is not semantically interpreted, and that the tense is interpreted as a bound variable. See von Stechow & Gronn (2013a, b) for a recent overview of the discussion. Even more recently, the conversation has been extended to include optional tense languages such as Washo (Bochnak, 2016) and Tlingit (Cable, 2017), in which the hypothesized LF structure is what surfaces in the language. The interested student is referred to those articles for further discussion of this issue.
13 | Modality

13.1 Introduction

*Note: This chapter is still in a preliminary stage.*

13.2 Opacity

The following might seem like a well-founded principle to adopt in everyday reasoning:

(1) **Leibniz’s law: Substitutability of identicals**

   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, the following argument appears valid:

   (2) a. Ruth Bader Ginzburg was the first woman to be on both *Harvard Law Review* and *Columbia Law Review*.

   b. No law firm in New York City agreed to hire Ruth Bader Ginzburg after she finished law school.

   c. Therefore, no law firm in New York City agreed to hire the first woman to be on both *Harvard Law Review* and *Columbia Law Review* after she finished law school.

   The conclusion seems to be licensed by the fact that Ruth Bader Ginzburg is the first woman to be on both *Harvard Law Review*
and Columbia Law Review. – they are the same person. Therefore any property that the one has, the other has as well.

But there are examples where Leibniz’s law does not hold. For instance, the Morning Star happens to be the Evening Star. (Both name the planet Venus, which, incidentally, is not a star.) Perhaps you did not know this already. Then someone could tell you (3a) and you would learn something. On the other hand, if someone were to tell you (3b)

(3)  a. The Morning Star is the Evening Star.
    b. The Morning Star is the Morning Star.

This would be entirely unhelpful (and perhaps rudely imply that you were not capable of sorting this out on your own). Although the Morning Star and the Evening Star refer to the same object, (3a) and (3b) do not express the same proposition. The latter expresses a necessary truth, a proposition that is true in all possible worlds. The former could be false in some possible world. 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)  a. Necessarily, the Morning Star is the Evening Star.
    b. Necessarily, the Morning Star is the Morning Star.

The reason for the non-substitutability of identicals in this case is that necessarily depends on the proposition expressed by the sentence it operates on, and not just on its truth value. This property renders necessarily unlike, say, negation, which depends only on the truth value; negation, as you may recall, is a ‘truth-functional’ connective. Whether or not a proposition necessarily holds depends on its truth value in every world, not just the world under consideration. In other words, necessarily depends on the intension of the sentence it combines with, and not just its extension. The extension of an expression is its semantic value at a particular world (so, for formulas, the extension is a truth value), while the
intension is a function from possible worlds to the extensions they have at those worlds. Expressions that depend on the intensions of the phrases they combine with, and not just their extensions, are called intensional.

Verbs expressing attitudes towards propositions (propositional attitude verbs) such as believe, know and want are also intensional, as they express attitudes towards propositions, and not just truth values. We find violations of Leibniz's law here as well. For example, Mary believes that the Morning Star is a planet does not imply that Mary believes that the Evening Star is a planet, as Mary might not know that the Morning Star is the Evening Star.

Some verbs of propositional attitude do not imply the truth of their complement. For example, (5a) does not imply (5b):

(5)  
\begin{enumerate} 
\item a. Mary believes that Fred is in Paris. 
\item b. Fred is in Paris. 
\end{enumerate}

Similarly, (6a) does not imply (6b):

(6)  
\begin{enumerate} 
\item a. Mary believes that a unicorn is eating her parsnips. 
\item b. A unicorn is eating Mary's parsnips. 
\end{enumerate}

In fact, (6a) does not even commit the speaker to the existence of unicorns, although it does imply that Mary believes that there are unicorns. In other words, (6a) lacks existential import with respect to the indefinite a unicorn.

Propositional attitudes can be embedded in transitive verbs that take a noun phrase direct object, and in such cases we observe the same effect. Thus, while (7a) implies that there is at least one unicorn, (7b) need not do so:

(7)  
\begin{enumerate} 
\item a. Andrea sees a unicorn. 
\item b. Andrea wants a unicorn. 
\end{enumerate}

To paraphrase Quine (1956), example (7b) can be interpreted to mean that Andrea merely seeks relief from unicorn-less-ness, not
that there is a particular unicorn that Andrea wants; no particular unicorn need even exist for the sentence to be true. Thus a representation of the following kind would not do:

$$\exists x. [\text{Unicorn}(x) \land \text{Wants}(a, x)]$$

because this implies that there are unicorns. How, then, should the meaning of a sentence like (7b) be represented? We will need to augment our representation language with tools for talking about other possible worlds in order to capture the meaning of these intensional expressions.

Before we embark on this task, however, we must observe that (7b) is ambiguous; it could be interpreted to mean that there is a particular unicorn that Andrea wants. The two readings involved here are called DE RE (‘of the object’) and DE DICTO (‘of the word’). On the de re reading, Andrea has a belief about a particular unicorn: Regarding that unicorn, she wants it. The de dicto reading is the one on which she merely seeks relief from unicorn-less-ness. In the latter case, the desire is not about a particular individual, rather it is about the category, unicorns; she wants that category to be instantiated in her possession. In this sense, the desire is about the ‘word’, naming the category.

Quine (1956), two whom the de dicto vs. de re distinction is usually attributed, illustrated the de dicto / de re ambiguity with the following example:

(8) Ralph believes that someone is a spy.

On the de re reading, Ralph has a belief about a particular object/individual: There is someone about whom Ralph believes that they are a spy. On the de dicto reading, Ralph has no particular individual in mind; he just believes that there are spies. The belief is not about a particular individual, rather it’s about the category, spies. Ralph believes that the category is instantiated. The de dicto interpretation does not commit the speaker to the existence of spies; only Ralph.
For another example:

(9) John believes that a Republican will win.

On one interpretation, there is a specific Republican who John believes will win. John may not even know that the person in question is a Republican. This is the de re interpretation. On the de dicto interpretation, there is no specific Republican that John believes will win; he just believes that whoever wins will be of that party.

Borrowing Quine’s metaphor, verbs like believe ‘seal off’ the complement clause. As a consequence, the existential import of the complement clause is not inherited by the sentence as a whole. In this sense, the verbs are opaque. (The opposite of opaque is transparent.) When an indefinite is interpreted within the scope of an opaque verb (i.e., with a de dicto reading), the sentence as a whole does not carry the existential import that the indefinite would normally produce.

**Exercise 1.** Give three more examples of opaque verbs and three more examples of transparent verbs. Be warned that there are some borderline cases, where intuitions are murky; do your best to stick to clear cases.

This kind of ambiguity occurs not only in philosophy texts, but also ‘in the wild’, or at least, in legal statutes. Anderson (2014) describes the following case.

In the fall of 2001, the accounting firm Arthur Andersen directed a large scale destruction of documents regarding its client Enron. Expecting a federal subpoena of records as a wave of accounting scandals unfolded, the firm urged its employees to begin shredding papers in October, shortly before the SEC began an official investigation into Enron. The shredding
ceased abruptly on November 9th, immediately on the heels of the SEC’s subpoena. In 2005, the Supreme Court reversed Arthur Andersen’s conviction for “knowingly . . . corruptly persuad[ing] another . . . with intent . . . induce any person to . . . withhold a record, document, or other object, from an official proceeding.” The conviction was defective in part because the jury instructions did not make clear that the defendant’s actions had to be connected to a particular official proceeding that it had in mind, which in this case had not been initiated at the time of the shredding. The ruling followed a line of obstruction of justice decisions dating back to the nineteenth century in holding that, if in its frenzy of paper shredding the defendant firm was not specific about the particular official proceeding to be obstructed, the statute could not have been violated.

On the *de re* interpretation (for the both the document and the proceeding from it) of the statute, it is violated when there is a particular document from a particular official proceeding which the perpetrator intends to withhold. On a *de dicto* interpretation, it is violated when the intent is such that there is an official proceeding from which documents are withheld. It is fair to say that Andersen would be guilty on a *de dicto* interpretation, and were acquitted on the basis of a *de re* interpretation.

**Exercise 2.** Consider the following case from [Anderson (2014)](Andersen):  

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 re-
luctantly. There had been voter fraud by impersonation, certainly. But the court fixated on the object of the impersonation and concluded that because a dead person could not vote, the defendant had not impersonated a “person entitled to vote.” The court attributed the mismatch between this result and the evident purpose of the statute to an oversight of the drafters: “The legislature has not used words wide enough to make the personation of a dead man an offence.”

How would you characterize the de re and de dicto interpretations, respectively, in this case? Which interpretation does the court appear to have taken? Is there an interpretation on which the man is guilty? Explain why or why not.

A good theory of propositional attitude verbs should be able to account for the de dicto vs. de re ambiguity. In order to build up the theoretical machinery necessary in order to do this, we will start with modal logic, which contains the intensional operators necessarily and possibly. We then present Montague’s Intensional Logic, which builds on modal logic and provides a mechanism for compositional interpretation of sentences involving intensional operators.

### 13.3 Modal Logic

#### 13.3.1 Alethic logic

As discussed above, (10a) expresses a truth that is not a necessary truth; things could have been otherwise, while (10b) is necessarily true.

10. a. The Morning Star is the Evening Star.
b. The Morning Star is the Morning Star.

In other words, the statement in (10a) is CONTINGENTLY TRUE, while the statement in (10b) is NECESSARILY TRUE.

We can also divide false statements into those that are necessarily false and those that are contingently false in an analogous manner. For example, both (11a) and (11b) are false, but the former is contingently false and the latter is necessarily false.

(11) a. Hillary Clinton is the president of the U.S.
    b. Hillary Clinton is not Hillary Clinton.

**Exercise 3.** Give another example of a contingently false statement and one example of a necessarily false statement.

A logical system representing concepts like *it is necessary that* and *it is possible that* is called an ALETHIC LOGIC or MODAL LOGIC. The term ‘modal logic’ is somewhat more common and frequent, but it also has a broader usage, sometimes also applying to tense logics of the Priorian kind. The operator ‘it is necessary that’ is standardly represented as a box, \( \Box \), and ‘it is possible that’ is represented as a diamond, \( \Diamond \). Thus in alethic logic the syntax rules are extended with the following:

- If \( \phi \) 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.\(^1\) The formalization here does not depend on a particular conception of possible worlds, but the authors tend to prefer Wittgenstein’s perspective.

The semantics of the modal operators can be defined syncategorically as follows:\(^2\)

\[ [\Box \phi]^{M,g,w} = 1 \text{ iff } [\phi]^{M,g,w'} = 1 \text{ for all } w' \]
\[ [\Diamond \phi]^{M,g,w} = 1 \text{ iff } [\phi]^{M,g,w'} = 1 \text{ for some } w' \]

It turns out that, using these definitions, certain intuitively valid sentences are indeed valid, for example:

- \(\Box \phi \leftrightarrow \neg \Diamond \neg \phi\)
  ‘It is necessarily the case that phi if and only if it is not possible that not phi’

- \(\Box \phi \rightarrow \phi\)
  ‘Necessarily phi implies phi’

- \(\phi \rightarrow \Diamond \phi\)
  ‘If phi, then possibly phi’

The first statement implies that \(\Diamond\) is the dual of \(\Box\). (In the same way, \(\exists\) is the dual of \(\forall\), since \(\forall x. \phi\) is equivalent to \(\neg \exists x. \neg \phi\).) In fact, sometimes in modal logic a statement of the semantics of \(\Diamond\) is left out, and the \(\Diamond\) is defined as a syntactic abbreviation of \(\neg \Box \neg\).

Given our semantics for the quantifiers from previous chapters, the following formulas are also valid:

\(^1\)See https://plato.stanford.edu/entries/possible-worlds/
\(^2\)This is one of many possibilities; this simplest system is known as S5. See Hughes & Cresswell [1968] for a fuller presentation.
• $\forall x \Box \phi \rightarrow \Box \forall x \phi$

• $\exists x \Diamond \phi \rightarrow \Diamond \exists x \phi$

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 word 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](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](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):

(12) My university email account no longer exists.
(13) If that existed, then I would have heard of it!

A famous example discussed by Russell (1905) is:

(14) The golden mountain does not exist.

This sentence is felt to be true; but in that case, what does the *golden mountain* refer to? One way of capturing these facts in a fixed-domain framework is to introduce an existence predicate *exists*, understood to be true of an individual at a world if that individual really exists at that world. Thus we can distinguish between two kinds of ‘existence’: BROAD EXISTENCE, which holds of everything a quantifier can range over, that is, everything in the domain of individuals, and NARROW EXISTENCE, which is a contingent property of individuals, holding at some worlds but not others. The verb *exist* can be taken to denote the narrow, contingent kind of existence, captured by the existence predicate.
13.4 Intensional Logic

13.4.1 Introducing Intensional Logic

In the previous section, we defined the semantics of $\Box$ and $\Diamond$ synchroncategorematically, rather than giving $\Box$ a meaning of its own. It is common to do this with negation as well. But with negation, unlike necessity, it is possible to give the symbol a meaning of its own, one that is a function of the truth value of its complement. The semantic value of the $\neg$ symbol can be defined as a function that returns 0 if it receives 1 as input, or 1 if it receives 0 as input. The same is not the case for $\Box$, because the truth of a $\Box$ statement depends not on the truth value of its complement at a particular world. We saw this above with Frege's examples, repeated here:

(15) Necessarily, the Morning Star is the Morning Star.
(16) 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 inten-
sion 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 \([\alpha]^{M,g,w}\). The intension of an expression \(\alpha\) is that function \(f\) such that \(f(w) = [\alpha]^{M,g,w}\). That function is sometimes denoted as follows:

\[[\alpha]^{M,g}_c\]

with the cent sign \(\_\) as a subscript on the denotation brackets, and no world variable superscript.

Note that it is not possible to figure out the intension from the extension at a particular world. In order to get the intension, you need to know the extension at *every* possible world. So there is no function from extensions to intensions. Note also that every expression in the language gets an extension, even variables. But since the denotation of a variable is always determined by an assignment function, its intension relative to \(g\) will be a function that yields the same value for every possible world given as input.

Now let us return to the problem of giving a compositional, non-syncategorematic semantics for necessity and belief. Recall that if the \(\Box\) operator denotes any function, it denotes one whose input is a proposition, rather than a truth value. The relevant
prophecy 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:

\[ ^\wedge \alpha \]

Relative to any given world, this expression has as its \textit{extension} the \textit{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:

\[ ^\wedge \text{Happy}(m) \]

has the intension of \( \text{Happy}(m) \) as its extension. (Put more simply: The extension of \( ^\wedge \text{Happy}(m) \) 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 $^\wedge \alpha$ 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 $^\wedge$ is as follows:

- If $\alpha$ is an expression of type $\tau$, then $^\wedge [\alpha]^{M,g,w}$ is that function $f$ with domain $W$ such that for all $w \in W$: $f(w)$ is $[\alpha]^{M,g,w}$.

The hat operator also has a partner, $^\wedge$ which moves from intensions to extensions. If $\alpha$ is an expression of type $\langle s, \tau \rangle$, then $^\wedge \alpha$ is an expression of type $\tau$. Its semantics is defined as follows:

- If $\alpha$ is an expression of type $\langle s, \tau \rangle$, then $^\wedge [\alpha]^{M,g,w}$ is the result of applying the function $[\alpha]^{M,g,w}$ to $w$.

With these tools in hand, let us now consider how we might get a handle on the de dicto / de re ambiguity and related puzzles. Let us introduce a constant $\text{bel}$, 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 de dicto reading of a sentence like John believes that a Republican will win can then be represented as follows:

$$\text{Bel}(\text{john}, ^\wedge \exists x[\text{Repub}(x) \land \text{Win}(x)])$$

The de re reading can be represented:

$$\exists x[\text{repub}(x) \land \text{bel}(\text{john}, ^\wedge \text{[win}(x)])]$$
In the latter formula, the existential quantifier and the predicate \texttt{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 the proper name \textit{Miss America}. The following sentence can be understood in two ways:

(17) John believes that Miss America is bald.

On the \textit{de re} interpretation, John might believe of some individual who happens to be Miss America without John knowing, that she is bald. On the \textit{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:

(18) \[ \lambda x. \text{Bel}(\text{john}, \neg \text{Bald}(x)) \](m)
(19) \text{Bel}(\text{john}, \neg \text{Bald}(m))

As the identity of Miss America varies from situation to situation, let assume that this name is \textit{not} a rigid designator, but rather a non-logical constant whose value can vary from world to world. Then, the first of the two formulas above captures the \textit{de re} interpretation; the second captures the \textit{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.

13.4.2  Formal fragment

Let us define a new logic, IL 'Intensional Logic', following Montague. The language is not exactly the same as Montague’s Intensional Logic, but it is fundamentally similar in spirit.

13.4.2.1  Semantics

The types are defined recursively as follows:

- \( t \) is a type
- \( e \) is a type
- If \( \sigma \) 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 the 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 \( \sigma \) 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 $(\sigma, \tau)$.

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 $\check{\alpha}$ 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 $\llbracket \alpha \rrbracket_{M,g,w} = I(\alpha, w)$.
   
   (b) If $\alpha$ is a variable, then $\llbracket \alpha \rrbracket_{M,g,w} = g(\alpha)$.

2. **Predication**
   
   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 $\llbracket \alpha = \beta \rrbracket_{M,g,w} = 1$ iff $\llbracket \alpha \rrbracket_{M,g,w} = \llbracket \beta \rrbracket_{M,g,w}$.

4. **Negation**
   
   If $\phi$ is a formula, then $\llbracket \neg \phi \rrbracket_{M,g,w} = 1$ iff $\llbracket \phi \rrbracket_{M,g,w} = 0$.

5. **Binary Connectives**
   
   If $\phi$ and $\psi$ are formulas, then:
   
   (a) $\llbracket \phi \land \psi \rrbracket_{M,g,w} = 1$ iff $\llbracket \phi \rrbracket_{M,g,w} = 1$ and $\llbracket \psi \rrbracket_{M,g,w} = 1$.
   
   (b) $\llbracket \phi \lor \psi \rrbracket_{M,g,w} = 1$ iff $\llbracket \phi \rrbracket_{M,g,w} = 1$ or $\llbracket \psi \rrbracket_{M,g,w} = 1$.
   
   (c) $\llbracket \phi \rightarrow \psi \rrbracket_{M,g,w} = 1$ iff $\llbracket \phi \rrbracket_{M,g,w} = 0$ or $\llbracket \psi \rrbracket_{M,g,w} = 1$.
   
   (d) $\llbracket \phi \leftrightarrow \psi \rrbracket_{M,g,w} = 1$ iff $\llbracket \phi \rrbracket_{M,g,w} = \llbracket \psi \rrbracket_{M,g,w}$.

6. **Quantification**
   
   (a) If $\phi$ is a formula and $\nu$ is a variable of type $a$ then $\llbracket \forall \nu. \phi \rrbracket_{M,g,w} = 1$ iff for all $k \in D_a$:

   $$\llbracket \phi \rrbracket_{M,g[\nu \rightarrow k],w} = 1$$
(b) If $\phi$ is a formula and $v$ is a variable of type $a$ then $\llbracket \exists v \cdot \phi \rrbracket^{\mathcal{M},g,w} = 1$ iff there is an individual $k \in D_a$ such that that:

$$\llbracket \phi \rrbracket^{\mathcal{M},g[v \mapsto k],w} = 1.$$ 

7. Lambda Abstraction

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

8. Alethic modalities (new!)

(a) $\llbracket \Box \phi \rrbracket^{\mathcal{M},g,w} = 1$ iff $\llbracket \phi \rrbracket^{\mathcal{M},g,w'} = 1$ for all $w'$

(b) $\llbracket \Diamond \phi \rrbracket^{\mathcal{M},g,w} = 1$ iff $\llbracket \phi \rrbracket^{\mathcal{M},g,w'} = 1$ for some $w'$

9. Intensionalization (new!)

If $\alpha$ is an expression of type $\tau$, then $\llbracket ^{\cdot} \alpha \rrbracket^{\mathcal{M},g,w}$ is that function $f$ with domain $W$ such that for all $w' \in W$: $f(w')$ is $\llbracket \alpha \rrbracket^{\mathcal{M},g,w'}$.

10. Extensionalization (new!)

If $\alpha$ is an expression of type $\langle s, \tau \rangle$, then $\llbracket ^{-} \alpha \rrbracket^{\mathcal{M},g,w}$ is the result of applying the function $\llbracket \alpha \rrbracket^{\mathcal{M},g,w}$ to $w$.

Exercise 4. Formulate a lexical entry for the adverb necessarily, and show how it accounts for the examples involving the Morning Star and the Evening Star.

Exercise 5. Formalize the de dicto and de re readings of the what the statute prohibits in the Andersen example on page 419 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 \( \hat{\} \)? Explain why or why not.

13.5 Fregean sense and hyperintensionality

Frege's assessment of his puzzle about identity built on a distinction between sense and reference. For Frege, 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 (20) it does not follow that (21) is true.

(20) Susan believes that \( 2 + 2 = 4 \).
(21) Susan 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 (22) does not imply (23) (example from Muskens 2005):

(22) Susan believes that the cat is in if the dog is out.
(23) Susan believes that the dog is in if the cat is out.

Both of these cases exemplify the PROBLEM OF LOGICAL OMNISCIENCE: in general, people do not believe all of the logical consequences of their beliefs. Phenomena in which the substitution of one expression for another that has the same intension leads to a difference in truth value are called HYPERINTENSIONAL. Such cases clearly show that the analysis of belief given in the previous section is inadequate, and moreover that a more fine-grained notion of meaning is required. Recent perspectives on the problem are collected in a special volume of the journal Synthese; see Jespersen & Duží 2015 for an overview.

But the existence of 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.

13.6 Explicit quantification over worlds

The astute reader may have noticed that in the system described in §13.4 there were no expressions of type $s$. This is partly because Montague and his contemporaries believed that there were no expressions that made reference to possible worlds. That assumption has since been challenged, and for that reason among others, it is generally preferred nowadays to use a formal system in which there is explicit quantification and binding of possible worlds. On this view, rather than writing

\[ \tilde{\text{Bald}}(m) \]

one writes rather:

\[ \lambda w . \text{Bald}_w(m) \]
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.

The *de dicto* vs. *de re* ambiguity in

(24) John believes that a Republican will win.

would be captured as follows. The *de dicto* reading would be translated as follows:

\[
\lambda w. \text{Bel}_w(j, \lambda w' \exists x[\text{Republican}_{w'}(x) \land \text{Win}_{w'}(x)])
\]

The *de re* reading would be translated thus:

\[
\lambda w. \exists x[\text{Republican}_w(x) \land \text{Bel}(j, \lambda w'. \text{Win}_{w'}(x))]
\]

Although the representation is a bit more cluttered, with world variables here and there, it does make explicit which world the various predicates hold in. In the *de dicto* formula, Republican is associated with \( w' \), the remote world, whereas in the *de re* reading, it is associated with \( w \), the main world of evaluation. This captures the fact that on the *de dicto* reading, there may be no particular Republican that John’s attitude relates to; indeed, the sentence could be true even if Republicans did not exist.

A famous example of this sort of system is Gallin’s (1975) Ty\(_2\), so called because it has two types (\( s \) and \( e \)) other than \( t \). The semantics literature still uses both styles, as each has its own advantages, although Gallin’s style is gaining in popularity.

### 13.6.1 Modal auxiliaries

So far we have discussed two types of modal expressions: the adjectives *necessary* and *possible*, denoting alethic modalities, and attitude verbs like *believe* and *hope*. We have not said anything about modal auxiliaries like *may*, *must*, *can*, and *have to*. For a more thorough and pedagogical discussion of this topic than what can be achieved here, the reader is encouraged to consult Chapter
3 of von Fintel & Heim 2011, which motivates a particular context-sensitive analysis of these elements.

### 13.7 Indexicals and Necessity

Let us now return to a puzzle concerning indexicals, now that we have a rudimentary treatment of intensional phenomena under our belts. Kaplan 1977 observed that the following sentence is always true, whenever uttered, and yet it does not express a necessary truth:

\[(25) \text{I exist.}\]

For most of us, anyway, it is far from necessary that we exist. Any number of circumstances could have conspired so that we never came into being. How can it be that this sentence is always true, yet not necessarily true?

Recall that in Kaplan’s theory, the extension of an expression depends on a context of utterance \(c\). Integrating this idea into our intensional semantics, the extension of an expression \(\alpha\) will depend on a model \(M\), and assignment \(g\), a possible world \(w\), and a context of utterance \(c\).

\[[\alpha]_{M,g,w,c}\]

The indexical constant \(i\) is defined as follows:

\[(26) \quad [i]_{M,g,w,c} = sp(c)\]

The context of utterance determines not only a speaker \(sp(c)\), an addressee \(ad(c)\), a time of utterance \(t(c)\), and a location of utterance \(l(c)\), but also a designated circumstance of evaluation \(w(c)\). The designated circumstance of evaluation \(w(c)\) is intuitively the world in which the utterance takes place. Truth in a context can then be defined as follows: An occurrence of \(\phi\) 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 an intensional model $M$ for a logic of indexicals would be a tuple:

$$M = \langle D, I, W, C \rangle$$

where $D$ is a set of individuals, $W$ is a set of possible worlds, $I$ is a world-relative interpretation function, and $C$ is a set of contexts. Now, there are certain constraints on these models. For example, the speaker of any context must be in the extension of the existence predicate exists at the world of the context $^3$ Formally:

$$(27) \text{ If } c \in C, \text{ then } sp(c) \in I_w(c)(\text{Exists}).$$

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 in Kaplan’s system.

But it is not a necessary truth. Kaplan’s logic of indexicals contains necessity and possibility operators defined in the standard way in modal logic. So $\square \text{Exists}(i)_{M,g,w,c} = 1$ 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 $\text{Exists}(i)$ is not a necessary truth.

$^3$Cf. conditions 10 and 11, p. 544 of Kaplan (1977).
Exercise 7. Explain why *I am not here now* 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?
Appendix

Let us take a moment to summarize what we have done. We are almost done with all of English, but not quite. Ha! There are extremely many topics which are fruitful to study from this perspective that we haven’t touched on at all:

- comparatives: prettier, more beautiful, more books, less pretty, fewer books, less milk
- superlatives: prettiest, most pretty, most books
- exclusives: only, sole(ly), exclusive(ly), mere(ly), just
- exceptives: except (for), save (that), but
- demonstratives: that glass over there
- questions: Did John kiss Mary? and embedded questions: John doesn't know whether he kissed Mary
- imperatives: Kiss Mary!

to name a few. And there is much remaining to be said about the topics we have touched on. However, the reader now has a starter kit. The following sections give the fragment of English that we have developed so far.
A.1 Logic: Partial typed lambda calculus (L₃)

Expressions of the following fragment of English given below will be translated into the following version of lambda calculus in which there are three truth values. Let us call the language L₃.

Types. e and t are types, and if a and b are types, then ⟨a, b⟩ is a type; nothing else is a type. For all type a, ⊤ₐ stands for the undefined entity of type a.

A.1.1 Syntax of L₃

The set of expressions of type a, for any type a, is defined recursively as follows. (An expression of type t is a formula.)

1. Basic expressions
   For each type a,
   (a) the non-logical constants of type a are they symbols of the form cₙ,ₐ for each natural number n.
   (b) the variables of type a are the symbols of the form vₙ,ₐ for each natural number n.

2. Predication
   For any types a and b, if α is an expression of type ⟨a, b⟩ and β is an expression of type a then α(β) is an expression of type b.

3. Equality
   If α and β are terms, then α = β is an expression of type t.

4. Negation
   If φ is a formula, then so is ¬φ.

5. Binary Connectives
   If φ and ψ are formulas, then so are ¬φ, [φ ∧ ψ], [φ ∨ ψ], [φ → ψ], and [φ ↔ ψ].
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 $a$ and $u$ is a variable of type $b$ then $[\lambda u . \alpha]$ is an expression of type $(b,a)$.

8. **Iota terms**
   If $\phi$ is a formula, and $u$ is a variable of type $a$, then $[\iota u . \phi]$ is an expression of type $a$.

9. **Definedness conditions**
   If $\phi$ is a formula, then $\partial(\phi)$ is a formula.

In addition, we have the following abbreviation conventions.
1. Square brackets that are outermost in an expression may be deleted.
2. An expression of the form $[[\phi \land \psi] \land \chi]$ or $[\phi \land [\psi \land \chi]]$ can be simplified to $[\phi \land \psi \land \chi]$. Similarly for disjunctions.
3. We may write $\pi(\alpha_1, ..., \alpha_n)$ instead of $\pi(\alpha_n)...(\alpha_1)$.
4. Brackets around a quantified formula can be dropped if it is rightmost (last) in a top-level expression, or rightmost in a larger constituent that ends in a bracket.
5. The dot may be dropped in a sequence of binders.
6. Square brackets that are immediately embedded inside parentheses can be dropped.

### A.1.2 Semantics of $L_3$

For each type $a$, there is an associated domain $D_a$. $D_e$ is the domain of individuals, $D_t$ is the set of truth values, and for any types $a$ and $b$, $D_{(a,b)}$ is the set of functions from $D_a$ to $D_b$.

Expressions are interpreted in $L_3$ with respect to both:
• a model $M = \langle D, I \rangle$ where $D$ is a non-empty set of individuals, and $I$ is a function assigning a denotation in $D_a$ to each non-logical constant of type $a$

• an assignment $g$, which is a function assigning to each variable of type $a$ a denotation from the set $D_a$

For every well-formed expression $\alpha$, the semantic value of $\alpha$ with respect to model $M$ and assignment function $g$, written $[\alpha]^{M,g}$, is defined recursively 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 $\langle a, b \rangle$, 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 $[-\phi]^{M,g} = 1$ iff $[\phi]^{M,g} = 0$.

5. **Binary Connectives**
   
   If $\phi$ and $\psi$ are formulas, then:
   
   (a) $[\phi \land \psi]^{M,g} = 1$ iff $[\phi]^{M,g} = 1$ and $[\psi]^{M,g} = 1$.
   
   (b) $[\phi \lor \psi]^{M,g} = 1$ iff $[\phi]^{M,g} = 1$ or $[\psi]^{M,g} = 1$.
   
   (c) $[\phi \rightarrow \psi]^{M,g} = 1$ iff $[\phi]^{M,g} = 0$ and $[\psi]^{M,g} = 1$.
   
   (d) $[\phi \leftrightarrow \psi]^{M,g} = 1$ iff $[\phi]^{M,g} = [\psi]^{M,g}$.

6. **Quantification**
(a) If $\phi$ is a formula and $v$ is a variable of type $a$ then $\left[ \forall v . \phi \right]_{M,g} = 1$ iff for all $k \in D_a$:

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

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

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

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

8. **Iota terms**
If $\phi$ is a formula and $u$ is a variable of type $a$ then:

$$\left[ iu . \phi \right] = \begin{cases} d & \text{if } k : \left[ \phi \right]_{M,g[u \mapsto k]} = 1 \\ \#e & \text{otherwise} \end{cases}$$

9. **Definedness conditions**
If $\phi$ is an expression of type $t$, then:

$$\left[ \partial(\phi) \right]_{M,g} = \begin{cases} \left[ \alpha \right]_{M,g} & \text{if } \left[ \phi \right]_{M,g} = 1 \\ \#a & \text{otherwise} \end{cases}$$

**Truth in a model.** For any expression $\phi$, $\left[ \phi \right]_M = 1$ iff $\left[ \phi \right]_{M,g} = 1$ for every value assignment $g$. Similarly, $\left[ \phi \right]_M = 0$ iff $\left[ \phi \right]_{M,g} = 0$ for every value assignment $g$.

### A.2 Syntax of English fragment

**Syntax rules.** The following rules derive trees at Deep Structure:

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

**Lexicon.** Lexical items are associated with syntactic categories as follows:

- **J:** *and, or*
- **Neg:** *it is not the case that*
- **V:** *smokes, loves, kissed, is*
- **A:** *lazy, proud*
- **N:** *drunkard, baby, kid, zebra, sister*
- **D:** *the, a, every, some, no, neither, 's, who, which John, Obama, everybody, somebody, nobody...*
- **P:** *of, with*
- **C:** *that*

**Transformations.** We assume the ‘T-model’, where a set of transformations convert Deep Structures to Surface Structures, Surface Structures to Phonological Forms, and Surface Structures to Logical Forms.
The only transformation from Deep Structure to Surface Structure that we will make explicit here is Relativization (cf. Muskens 1996):

**Relativization (DS \( \rightarrow \) SS).** If \( \alpha \) is *who*, *whom* or *which*:

\[
[s \ X [dp \ d \ \alpha ] \ ] Y \Rightarrow [cp \ \alpha_i \ [c' \ [\emptyset ] \ [s \ X [dp \ t_i ] \ ] Y ] ]
\]

where \( i \) is a fresh index.

The structures that are interpreted are Logical Forms, which are derived from Surface Structures using Quantifier Raising (QR). Following May (1985), we assume that QR only allows adjunction to S nodes (whereas Heim & Kratzer (1998) allow adjunction to any expression of an appropriate semantic type), but we take the insertion of a numerical index into the tree from Heim & Kratzer (1998).

**Quantifier Raising (SS \( \rightarrow \) LF).**

\[
[s \ X [dp \ \alpha ] \ ] Y \Rightarrow [ [dp \ \alpha ] \ [\lambda p \ i \ [s \ X [dp \ t_i ] \ ] Y ] ]
\]

where \( i \) is a fresh index.

### A.3 Translations

#### A.3.1 Lexical entries

We associate each lexical item with a translation to \( L_3 \). We will use the following abbreviations:

- \( x \) is \( v_{0,e} \), \( y \) is \( v_{1,e} \), and \( z \) is \( v_{2,e} \).
- \( X, Y, P \) and \( Q \) are variables of type \( \langle e, t \rangle \).
- \( R \) is a variable of type \( \langle e, \langle e, t \rangle \rangle \).
- \( p \) and \( q \) are variables of type \( t \).
- \( b, l, m, h \) and \( r \) are constants of type \( e \).
• drunkard, baby, kid, zebra, lazy, and snores are constants of type \(\langle e, t \rangle\).

• loves, kissed, with, proud, and sister are constants of type \(\langle e, \langle e, t \rangle \rangle\).

Type \(\langle e, t \rangle\):

1. \(\langle drunkard \rangle = \lambda x.\text{drunkard}(x)\)

2. \(\langle baby \rangle = \lambda x.\text{baby}(x)\)

3. \(\langle kid \rangle = \lambda x.\text{kid}(x)\)

4. \(\langle zebra \rangle = \lambda x.\text{zebra}(x)\)

5. \(\langle lazy \rangle = \lambda x.\text{lazy}(x)\)

Type \(e\):

1. \(\langle Homer \rangle = h\)

2. \(\langle Maggie \rangle = g\)

3. \(\langle Bart \rangle = b\)

4. \(\langle Lisa \rangle = l\)

5. \(\langle Marge \rangle = m\)

Type \(\langle t, \langle t, t \rangle \rangle\):

1. \(\langle \text{and} \rangle = \lambda p\lambda q.\left[p \land q\right]\)

2. \(\langle \text{or} \rangle = \lambda p\lambda q.\left[p \lor q\right]\)

Type \(\langle t, t \rangle\):

1. \(\langle \text{it is not the case that} \rangle = \lambda p.\neg p\)

Type \(\langle \langle e, t \rangle, \langle e, t \rangle \rangle\):
Modality

1. \( \langle \text{is} \rangle = \lambda P. P \)
2. \( \langle a \rangle = \lambda P. P \)

Type \( \langle (e, t), e \rangle \):

1. \( \langle \text{the} \rangle = \lambda P. \, \, t \, \, x \, \, . \, \, P(\, x) \)

Type \( \langle e, \{e, t\} \rangle \):

1. \( \langle \text{loves} \rangle = \text{loves} \)
2. \( \langle \text{kissed} \rangle = \text{kissed} \)
3. \( \langle \text{with} \rangle = \text{with} \)
4. \( \langle \text{proud} \rangle = \text{proud} \)
5. \( \langle \text{sisiter} \rangle = \text{sisiter} \)

Type \( \langle (e, t), t \rangle \):

1. \( \langle \text{something} \rangle = \lambda P. \, \, \exists \, \, x \, \, . \, \, P(\, x) \)
2. \( \langle \text{nothing} \rangle = \lambda P. \, \, \neg \, \, \exists \, \, x \, \, . \, \, P(\, x) \)
3. \( \langle \text{everything} \rangle = \lambda P. \, \, \forall \, \, x \, \, . \, \, P(\, x) \)

Type \( \langle (e, t), \{e, t\} \rangle \):

1. \( \langle \text{some} \rangle = \lambda P \lambda Q. \, \, \exists \, \, x \, \, . \, \, [P(\, x) \, \, \land \, \, Q(\, x)] \)
2. \( \langle \text{no} \rangle = \lambda P \lambda Q. \, \, \neg \, \, \exists \, \, x \, \, [P(\, x) \, \, \land \, \, Q(\, x)] \)
3. \( \langle \text{every} \rangle = \lambda P \lambda Q. \, \, \partial[\, \, \exists \, \, x \, \, [P(\, x)]\, \, \land \, \, \forall \, \, x \, \, [P(\, x) \, \, \rightarrow \, \, Q(\, x)] \)
4. \( \langle \text{neither} \rangle = \lambda P \lambda Q. \, \, [\, \, \partial[\, |P| = 2\, \, \land \, \, \neg \, \, \exists \, \, x \, \, [P(\, x) \, \, \land \, \, Q(\, x)] \, \, ] \)

Type \( \langle e, e \rangle \):

1. \( \langle \text{of} \rangle = \lambda x. \, \, x \)
A.3.2 Composition rules

If the translation of an expression $\gamma$ is not specified in the lexicon, then it is given by one of the following rules:

1. **Functional Application**
   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:
   $$\langle \gamma \rangle = \langle \alpha \rangle (\langle \beta \rangle)$$

2. **Predicate Modification**
   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 $\langle \beta \rangle$.

3. **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 $\langle \gamma \rangle = \lambda v_{i,e} . \langle \beta \rangle$.

4. **Pronouns and Traces**
   If $\alpha$ is an indexed trace or a pronoun, $\langle \alpha_i \rangle = v_{i,e}$

5. **Non-branching Nodes**
   If $\beta$ is a tree whose only daughter is $\alpha$, then $\langle \beta \rangle = \langle \alpha \rangle$.

We also have the following type-shifting rules:

1. **Predicate-to-modifier shift (MOD)**
   If $\langle \alpha \rangle$ is of category $\langle e, t \rangle$, then:
   $$\langle \text{MOD } \alpha \rangle = \lambda P \lambda x . [\langle \alpha \rangle (x) \land P(x)]$$
   as well (as long as $P$ and $x$ are not free in $\langle \alpha \rangle$; in that case, use different variables of the same type).
2. **Argument Raising**
   If an expression has a translation $\alpha$ of type $\langle \overrightarrow{a}, \langle b, \langle \overrightarrow{c}, t \rangle \rangle \rangle$, then that expression also has translations of the following form:
   \[
   \lambda \overrightarrow{x} \overrightarrow{a} \lambda v_{\langle b, t \rangle} \lambda \overrightarrow{y} \overrightarrow{c} \cdot v(\lambda z_{b}[\alpha(\overrightarrow{x})(z)(\overrightarrow{y})])
   \]

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).

4. **Iota shift**
   If $\llangle \alpha \rrangle$ is of type $\langle e, t \rangle$, then
   \[
   \llangle \text{IOTA } \alpha \rrangle = \iota x. \llangle \alpha \rrangle(x)
   \]
   as well (unless $x$ is free in $\alpha'$; then choose a different variable).
Bibliography


Cable, Seth. 2008. Tense, aspect and aktionsart. Lecture notes, Theoretical Perspectives on Languages of the Pacific Northwest, Proseminar on Semantic Theory.


Gallin, Daniel. 1975. *Intensional and higher order modal logic*. Amsterdam: North Holland Press.


Hackl, Martin. 2009. On the grammar and processing of proportional quantifiers: most vs. more than half. Natural Language Semantics 17. 63–98. https://doi.org/10.1007/s11050-008-9039-x


Heim, Irene. 1994. Comments on Abusch’s theory of tense. Ms., MIT.


Oliver, Alex & Timothy Smiley. 2013. *Plural logic*. Oxford University Press.


