

3

Compositionality, Direct Compositionality, and the syntax/semantics interface

3.1. Building a fragment: First steps	43	3.3. Folding in worlds (and times)	48
3.2. Implicatures vs truth conditions	46	3.4. Negation: A first pass	49

As noted in the preceding chapter, the program of semantics often takes as its point of departure the adage “To know the meaning of a (declarative) sentence is to know what it would take to make it true.” There is a second foundational adage: “The meaning of the whole is computed in some predictable way from the meaning of its parts.” This is also often known as *Frege’s principle* or *the principle of compositionality*. Taken in its broadest sense, this is fairly weak: obviously speakers of a language know the basic units (words, or more accurately, morphemes) and know some system to combine these into larger meaningful expressions. Surely at least most of language has to be compositional, or there would be no way to systematically combine units into larger meaningful expressions which have never been uttered (or heard) before.¹ In other words, we can see the grammar of a natural language as a specification of the set of well-formed expressions of the language, combined with rules which map each well-formed expression

¹ But there is a question about whether everything is compositional. Idioms are a classic case of items that at least appear to be non-compositional.

into a meaning (that is, some object built out of our model-theoretic toolbox, which so far consists only of truth values, worlds, and times).

But while it is generally accepted that the meanings of larger expressions are computed in some predictable way from the meanings of smaller ones, what is not so clear is just what that predictable way is and how complex it is. A very simple and elegant view on this is as follows. The syntax consists of statements that predict the existence of well-formed expressions on the basis of the existence of other well-formed expressions. At the base of this is the lexicon; the list of basic units (for our purposes, words). The semantics works in tandem with the syntax: each syntactic rule which predicts the existence of some well-formed expression (as output) is paired with a semantic rule which gives the meaning of the output expression in terms of the meaning(s) of the input expressions. This is what we mean by **Direct Compositionality**, and is the point of view that will mainly be pursued in this book.

3.1. Building a fragment: First steps

Every expression of a language, including the basic expressions (the words), can be seen as a triple of <[sound], syntactic category, [[meaning]]>. Borrowing notation from phonology, we enclose the sound part in square brackets (although we simply use English orthography rather than phonetic transcription), and the meaning part is enclosed in double square brackets [[...]]. A rule is thus something which takes one or more triples as input and yields a triple as output. A reader who is used to thinking of the syntax as consisting not of a number of very specific “rules” (such as phrase structure rules) but a set of much more general rules (or “principles”) such as the X-bar schemata need not worry: Chapter 6 will give the rules in far more general form.² Since we will often be adopting rules which are temporary and will later be revised, simplified, or generalized, any rule not part of the

² A certain amount of modern linguistics has eschewed the term “rule” on the grounds that the grammar hopefully does not consist of a list of a large number of “rules” rather than having the apparatus stated in much more general format. Nonetheless, most agree that the grammar contains statements that predict the set of well-formed expressions and assign them a meaning, and here the term “rule” is being used in the most general sense here to simply mean these statements. We will

final fragment will be notated as TR (followed by a number). (Rules which do remain in the ultimate fragment are simply labeled with R.)

Assume that the set of syntactic categories includes S which we use to mean a sentence.³ We continue here to assume that all Ss have as their truth value 1 or 0 (i.e., none are undefined in some worlds). Moreover, for now let us go back to ignoring the world (and time) parameters, and give a purely extensional semantics. Then we begin with the following:

TR-1. If α is an expression of the form $\langle [a], S, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], S, [[\beta]] \rangle$, then there is an expression γ of the form $\langle [\alpha\text{-and-}\beta], S, 1 \text{ if and only if } [[\alpha]] = 1 \text{ and } [[\beta]] = 1 \rangle$. (Hereafter we use the standard *iff* to abbreviate “if and only if.”)

There are a variety of other ways this rule could be written. For example, we could write the syntax as a (*context-free*) *phrase structure rule* and pair this with a semantic part. This notation was used in much of the literature within Generalized Phrase Structure Grammar (GPSG) (Gazdar, Klein, Pullum, and Sag 1985):

TR-1'. $S_1 \rightarrow S_2 \text{ and } S_3; \quad [[S_1]] = 1 \text{ iff } [[S_2]] = 1 \text{ and } [[S_3]] = 1.$

A phrase structure rule by itself, then, is a rule which gives the phonological and syntactic part of a rule such as TR-1 but not the semantics. When supplemented with the semantics—as in TR-1'—this is just TR-1 in a different format. Note also that TR-1' uses subscripts in the syntactic part of the rule, but these are not actually meant as a part of the syntactic category.⁴ These are needed just to state the semantics; this is avoided in the notational system in TR-1.

indeed be moving towards a view of grammar which contains just a few rather general statements to this effect, but there is no reason not to call these rules.

³ In some current theories of syntax (Minimalism and some of its predecessors) this category is instead called TP (*Tense Phrase*) and in slightly earlier versions of that theory it is called IP (*Inflectional Phrase*). As with the label NP, we are using the terminology standard in many other theories and in many other related disciplines; readers familiar with the terms TP or IP instead can make the appropriate translation.

⁴ A *context-free phrase structure rule* specifies the well-formedness of an expression of some grammatical category X in terms of concatenations of strings of other expressions in the language. These other expressions are described either listed on a case-by-case basis (like *and* above) or are described by their grammatical category. The set of grammatical categories is finite, and it does not include things like S_1 etc.

To forestall one objection: one should not be misled by the fact that the word “and” is used in the semantic part to give a meaning for English sentences containing *and*. The word “and” in the first part of the triple in TR-1 is the English word [ænd] where English is the *object language*—the language whose syntax and semantics is being modeled. But we need a language in which to describe the grammar, and we have chosen English for that purpose as well. So here English is also being used as the *metalanguage*. The confusion that arises by using English as the metalanguage to model English as the object language would go away if we were using English to, e.g., model the grammar of Swahili—or if we wrote the semantics for English using some sort of purely symbolic notation (whose meaning had been agreed on). There are other ways to alleviate the potential confusion; we could just as well have written the semantic part of the rule by means of what is known as a truth table. This is illustrated below where we rewrite the semantic part of TR-1 by listing the possible combinations of values for each of the α and β followed by the value that this gives for the string $[\alpha\text{-and-}\beta]$:

(1) $[[\alpha]]$	$[[\beta]]$	$[[\alpha\text{-and-}\beta]]$
1	1	1
1	0	0
0	1	0
0	0	0

From now on we use the format in TR-1 as our official notation, although at times will use phrase structure rule format or other notational systems as convenient. Incidentally, TR-1 introduces *and* directly as part of the rule rather than treating it as an item in the lexicon with a grammatical category. This is known as treating it *syncategorematically*. This decision to treat *and* not as a lexical item but rather as only something introduced by a rule is not essential (and is almost certainly incorrect), and will be revised later.

The rule for *or* is analogous (here we will highlight the lack of need to use the English word “or” in stating the semantics).

(Of course we could posit such a category, as long as there are a finite number of them, but that is not the intent in the rule above.) In the rule above, the idea is that the syntactic rule is just $S \rightarrow S$ and S . The reason that the subscripts are there in the rule is because the semantics needs to refer to them. Note that this is just an artifact of this particular notational system; the rule in TR-1 doesn’t use any subscripts in the syntax.

- TR-2. If α is an expression of the form $\langle [\alpha], S, [[\alpha]] \rangle$ and β is an expression of the form $\langle [\beta], S, [[\beta]] \rangle$, then there is an expression γ of the form $\langle [\alpha\text{-or-}\beta], S, 0$ iff $[[\alpha]] = 0$ and $[[\beta]] = 0 >$.

3.2. Implicatures vs truth conditions

Before continuing with the fragment, a digression is in order about the meaning of *or*. (More extensive discussion is found in section 10.4.) According to the semantics in TR-2, the sentence in (2) is true if Carol skydives and doesn't bungee jump, if she bungee jumps and doesn't skydive, and also if she does both.

- (2) Carol will go sky dive on her vacation, or she will go bungee jumping on her vacation.

This is what is known as *inclusive or*: both sentences can be true. But is that result correct? One might at first think not—there surely is a strong temptation to read (1) as saying she will do one or the other but not both. On the basis of this it is tempting to think that English *or* actually has as its meaning what is known as *exclusive or*—either conjunct could be true but not both. If that were correct then of course TR-2 needs revision.

But we can show that English *or* is actually inclusive. Although we often take sentences like (2) to convey that both events will not happen, this is not a part of its actual truth conditions. Note first that if I say (2) to you and it turns out that Carol goes completely wild and does both, you can hardly accuse me of lying. And it is easy to construct cases where there is no exclusive *or* suggestion and where things would not make sense if the meaning of *or* were “one or the other but not both.” So, take a linguistics department which is very strict about enforcing prerequisites. There is a course called Phonetics and Phonology II (Phon/Phon II). The course brochure contains the language in (3):

- (3) Students may not enroll in any course unless they have had the prerequisite(s).
You will be allowed to enroll in Phon/Phon II if and only if you have taken Phonology I or you have taken Phonetics I.

Suppose you are a student in love with all of phonetics and phonology, so of course you have taken both Phonetics I and Phonology I. You now go to enroll in Phon/Phon II. Surely you would be shocked if the Registrar told

you could not enroll in this course. Anyone reading this course brochure would certainly come to the conclusion that someone who had both courses is eligible for the more advanced one. But if *or* meant “one or the other but not both” then that is exactly what (3) would be saying.⁵

Of course there is still one other possibility: that *or* is ambiguous and that (2) uses “exclusive *or*” and (3) uses “inclusive *or*.” But if it were ambiguous, then each of these sentences should be too—it should be possible to read (3) in such a way that the Registrar would be entitled to block you from enrolling. But (3) has no such understanding; the Registrar simply could not legitimately come back to you and say “Oh, sorry you misunderstood. You’ve taken both Phonetics I and Phonology I; so you do not fit the eligibility requirement for enrollment in Phon/Phon II.”⁶

⁵ This point is amusingly illustrated by the following true story. An Ivy League institution which will remain nameless has a teaching center as a resource for graduate students to hone their teaching skills, and gives a certificate to any student who completes a program that includes a series of workshops and other activities (including a “micro-teaching” session). In February of 2011, students in the program received the following e-mail:

Attendance in Workshop #5 is MANDATORY if you belong in any of the following categories:

- a. You haven’t completed workshop #5, or
- b. You haven’t completed the Micro-Teaching requirement, or
- c. You haven’t completed workshop #5 AND you haven’t completed the Micro-Teaching requirement.

Of course anyone reading this would be completely amused and wonder why on earth (c) was included here. Surely, no one who had both not completed the workshop and not completed the Micro-Teaching requirement would really think that they were exempt. The fact that we react to this e-mail as being a silly instance of bureaucratic language confirms the point.

⁶ The argument is not quite complete, because one might still think that *or* is ambiguous, but that the reason the exclusive interpretation does not emerge is simply because it would be strange in this situation. Our knowledge of how prerequisites work makes this a highly unlikely interpretation in this scenario. But we can tweak the scenario to make the “exclusive” interpretation one that *might* make sense, but is still absent. Suppose that the linguistics department is also very unhappy with students taking courses that overlap too much with what they already know. And suppose that Phon/Phon II overlaps in some ways with Phonetics I and in some ways with Phonology II. In other words, the department is concerned not only about underqualified students, but also overqualified ones. Yet even in this situation, you would be genuinely outraged if the Registrar blocked your enrolment in the course.

Thus while an example like (2) often carries with it the suggestion that only one of the two conjuncts is true, this is just a suggestion and is what is known as an *implicature* rather than being part of the truth conditions. Sometimes the exclusive implicature is quite natural, as in (2), and sometimes it disappears, as in (3). In fact, when embedded under *if* (as in (3)) the exclusive suggestion associated with *or* systematically disappears. Ultimately one would hope for a single, stable meaning, where other principles can predict when the exclusive *or* implicature arises and when it doesn't. Indeed, by assigning it the inclusive semantics in TR-2 combined with a theory of language use (pragmatics), this can be done. (Many readers will be familiar with this under the rubric of *scalar implicature*.) This is the subject of sections 10.4 and 16.6.

3.3. Folding in worlds (and times)

TR-1 and TR-2 are obviously inadequate in that they are given purely *extensionally*: they both ignore the fact that the value of a sentence is not 1 or 0 but a function of type $\langle w, t \rangle$ (or really $\langle s, t \rangle$). For convenience, this book generally gives the rules in purely extensional fashion, but this is always just for expository simplification. We can easily write these rules in a more adequate fashion. Thus let us rewrite TR-1 as what we will call INT-TR-1 (INT here for intensional) as follows (actually this includes just the world argument and not the time; the full intensional version would replace “w” with “s” (a world–time pair):

INT-TR-1. If α is an expression of the form $\langle [a], S, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], S, [[\beta]] \rangle$, then there is an expression γ of the form $\langle [a\text{-and-}\beta], S, \text{for any world } w, [[\gamma]](w) = 1 \text{ iff } [[a]](w) = 1 \text{ and } [[\beta]](w) = 1 \rangle$.

Very often the world argument is given as a superscript, so one can rewrite the semantic part of this rule as given below:

$$[[\gamma]]^w = 1 \text{ iff } [[a]]^w = 1 \text{ and } [[\beta]]^w = 1$$

Indeed, if the intent were to keep out both the underqualified and the overqualified student, the brochure would have to be revised to say “if you have taken Phonology I or you have taken Phonetics I and you have not taken both.” So even in a scenario designed to bring out the “exclusive *or*” interpretation, it is not there.

One will commonly find this notation in the literature (and this will be used from time to time in this book). Notice that $[[a]]^w$ means exactly the same thing as $[[a]](w)$ —the superscript notation has an implicit “for all worlds w ” at the beginning of the whole rule. One should keep in mind that nothing (other than taste and expository ease) hinges on the choice of notation. The grammar is a system that proves expressions well-formed and assigns each expression a model-theoretic interpretation; the notation that one chooses to express the model-theoretic object being assigned is of no theoretical consequence.

- 3.1.** Rewrite the intensional version of TR-2 (i.e., INT-TR-2) using both of the notations for dealing with the world argument shown above.
- 3.2.** For any function f that characterizes a set, let us use the notation f_s to mean the set characterized by f . Now recall that we can think of the value of a sentence as a function of type $\langle w, t \rangle$ (a function from worlds to truth values) or—equivalently—as a set of worlds; the set mapped to 1 by the function. Thinking in set terms, what does *and* do? (That is, what operation on sets does it correspond to?) What about *or*? Using the notation suggested just above, write out the semantics for INT-TR-1 in set terms. Do the same for INT-TR-2.

3.4. Negation: A first pass

Readers familiar with first-order logic (or simple propositional logic) will recognize that the syntax and semantics of English sketched so far looks a bit like the syntax of simple logical languages in which two propositions p and q may be connected with \wedge (“and”) and \vee (“or”). A third operator that one might be used to seeing from elementary logic is a negation operator which is prefixed in front of a proposition. (Thus, in the development of an elementary propositional logic, there is a rule saying that if p is a well-formed formula then so is $\sim p$. The truth value of $\sim p$ is the reverse of that of p .)

If we try to model English directly using the tools from first-order logic in this way, we immediately reach a challenge for the Direct Compositional program. For a sentence like (4b) is the negation of (4a): if the first is true the second is false and vice versa:

- (4) a. Olympia Snowe voted for Bill 1838 (the only time it came up for a vote).
 b. Olympia Snowe didn't vote for Bill 1838 (the only time it came up for a vote).

Yet the syntax of English is completely different from the syntax of propositional logic. *Not* is not prefixed at the beginning of a sentence. It is not even prefixed at the beginning of a verb phrase. It generally shows up as a clitic on an auxiliary (as in *didn't* above, where it is a suffix attached to *did*.) Semantically, though, it seems reasonable to think of it as negating the entire sentence in which it sits.

There are two choices for resolving this apparent mismatch between the syntax and the semantics. One is to reject the Direct Compositional hypothesis. A common solution here is to imagine that actual English syntax is mapped into another level (called Logical Form, or LF) and that the grammar proceeds containing a rule or series of rules that map (4) into something like (5) (or perhaps something even more abstract), and that the compositional semantics interprets (5):

- (5) not [Olympia Snowe voted for Bill 1838 (the only time it came up for a vote)]

Then *not* here can be given the interpretation exactly like the symbol \sim is in logic. The second possibility is to conclude that it is a mistake to try to copy logic in giving a meaning for English *n't*—its meaning might have something to do with “ \sim ” but perhaps it is a more complex packaging of negation. This is the strategy that we will ultimately aim for; it allows for a much simpler conception of the grammar.

Temporarily, though, we take a third tack: which is to postpone incorporating into our fragment the natural way to do negation in English, and instead pretend that English has a single word *it-is-not-the-case-that* which can be prefixed to a sentence to negate it. Hence we enrich our fragment with TR-3 and its intensional counterpart INT-TR-3:

- TR-3. If α is an expression of the form $\langle [a], S, [[a]] \rangle$ then there is an expression β of the form $\langle [\text{it-is-not-the-case-that-}\alpha], S, 1 \text{ iff } [[a]] = 0 \rangle$.

INT-TR-3. If a is an expression of the form $\langle [a], S, [[a]] \rangle$ then there is an expression β of the form $\langle [\text{it-is-not-the-case-that-}a], S, [[\beta]] \rangle$ is a function from worlds to truth values such that for any w , $[[\beta]](w) = 1$ iff $[[a]](w) = 0$. (or, using the shorthand, the semantic part would read $[[\beta]]^w = 1$ iff $[[a]]^w = 0$).

For truth in advertising, while the final fragment will be closer to actual English, it will still not be a complete account of English *n't*, simply because space precludes an account of the English auxiliary system. But TR-3 will be improved upon.

3.3. As in Exercise 3.2, rewrite this in set terms. What is the operation on sets that corresponds to negation?

4

Expanding the fragment: Syntactic categories and semantic types

4.1. Noun phrases	52	4.4. Back to the compositional semantics	62
4.2. Intransitive verbs/verb phrases	58		
4.3. A brief look at lexical semantics	60	4.5. Illustrating the syntactic/semantic composition	63

4.1. Noun phrases

English has a variety of expressions like *the woman who is jogging down Blackstone Boulevard*, *the most disobedient puppy in Alexandra's puppy class*, *the house on the southeast of Chace Ave. and Hope St.*, etc. We will take it as given that these are all of the same grammatical category and will refer to them as NPs (see Chapter 1, n. 8). Let us assume that all expressions of the same syntactic category correspond to the same type (very broadly speaking) of semantic object. This assumption was already made for the case of S; we took each S to have a meaning of type $\langle s, t \rangle$, i.e., a function from world–time pairs to truth values. Recall that our basic semantic toolbox so far consists of the set of truth values, worlds, and times. But none of these are obvious candidates for NP meanings. What, then, is the type of thing that NPs have as their semantic values?

For the NPs illustrated above the answer seems simple: add to the toolbox a set of individuals, and assume that each NP picks out an individual. The notation e will denote the set of individuals; hence the value of an NP is some member of e . Of course, one immediately observes that this has a problem analogous to the problem of saying that the value of a sentence is a truth value. After all, the individual picked out by *the most disobedient puppy in Alexandra's puppy class* is entirely contingent on how the world happens to be: it could have been the puppy Mitka but it could instead have been the puppy Brunhilde. Besides, this can also depend on when this is uttered—the referent of the puppy-NP will change from year to year. Moreover, one can know the meaning of an NP without knowing who it picks out: I know what it means to say *the tallest man in the world on Jan. 1, 2011* but I have no idea who this is. So the solution is also analogous to the situation with sentences: the *intension* (the actual meaning) of an NP is a function from worlds and times to individuals—that is, some function in $\langle s, e \rangle$. Note that this says that if one knows the meaning of an NP *and knows everything there is to know about the actual world and time*, one can indeed pick out the relevant individual. But we are not omniscient, so we know the meaning (the intension) without necessarily knowing the extension. Our toolbox for constructing meanings now consists of the set of worlds (w), the set of times (i) (we continue to use s for the set of world–time pairs), the set of truth values (t), and the set of individuals (e). Incidentally, one might wonder whether the individuals are separate from the worlds: does it make sense to talk about an individual independent of the world in which it is located? As for the case of times, we will here treat the set of individuals as separate from the set of worlds and the evidence for this is quite similar (we can march an individual from world to world using a counterfactual conditional). We return to this shortly.

The claim that NPs pick out (in any given world and time) an individual has some interesting consequences. First, this means we need to think of “individual” rather broadly; the domain of individuals needs to include quite abstract things like those things picked out by the NPs *the idea that the earth is flat*, *the desire to leave*, etc. Moreover, in addition to our normal individuals like you, me, and my dog we have plural individuals (*the dogs*). We will not deal with these here, save to say that they can be treated as a special kind of individual (Link 1983). Finally, there is a class of expressions traditionally called NPs which really do seem to resist being treated as individuals: these are “NPs” with quantificational determiners like *every*

dog, *no dog*, *few dogs*, and so forth. These will simply be excluded from the fragment for now. They form the basis of some of the most interesting and rich work in semantics, and will be the subject of Chapter 10.

Consider now the case of names like *Mitka*, *Barack Obama*, etc. Traditional grammar tells us that these are “nouns” (they are considered “proper nouns” as opposed to “common nouns” like *dog*, *table*, *stream*, but both are called nouns). But in fact names (in English) have nothing in common syntactically, morphologically, or semantically with common nouns. Unlike ordinary common nouns, they do not take plural morphology. They do not occur with determiners like *the*.¹ And, as we will see below, while NPs denote individuals, common nouns surely do not (*the hungry dog* picks out an individual, but *dog* does not). On the other hand, names have exactly the same syntactic distribution as do ordinary NPs; the name *Mitka* can occur in all the same syntactic environments as an NP like *the disobedient husky*. Moreover, names and NPs are similar semantically: clearly a name picks out an individual, just like more a complex NP containing *the*. Given that names have the same syntactic distribution and the same semantic type as NPs in general, the conclusion is obvious: they are NPs. Our fragment will treat these as listed in the lexicon as NPs. Incidentally, we will also for now take complex NPs such as *the disobedient husky*, *the first man on the moon*, and so forth to just be listed in the fragment as single items; the internal structure of these is the subject of Chapter 8.

Note that this departs from a traditional view that a category label like NP (noun *phrase*) is an appropriate label only for expressions containing more than one word. Along with this traditional view goes the idea that single words must have as their category one of what is traditionally called the “basic parts of speech”—noun, verb, adjective, preposition, determiner (or article), adverb, and perhaps a few others. (These are the “parts of

¹ There are examples where proper nouns occur with determiners, as in (i) and (ii):

- (i) The Berrymans are coming over to dinner tonight.
- (ii) Every Dmitri in the semantics class might be a bit annoyed at the use of the name *Mitka* in so many example sentences.

But we can analyze these as cases where a proper noun shifts its class and is used as an ordinary common noun. English is very free with shifting items from one category to another (along with a meaning change). It should, however, be noted that a number of semanticists have explored a different view of proper names according to which *Mitka* instead means something like “the one who is named Mitka.”

speech” that still form the subject of colorful posters decorating elementary school walls, along with such odd definitions as “a verb is an action word” and “an adjective describes a pronoun.”) The traditional parts of speech are not entirely fiction—the notion that at least some of these categories are appropriate labels for a class of words is based on *morphological commonalities*. “Nouns,” for example, can (generally) combine with the suffix */z/* to form plurals; “verbs” combine with (among others) the suffix *-ing*, “adjectives” can (generally) combine with *-er* to form comparatives, etc. Thus the grammar may well need access to the notion of a class of “nouns,” “verbs,” and “adjectives” for the purposes of the morphology. But even granted that, we have already seen that proper nouns do not show noun-like morphology. There is no reason then not to call them NPs. The category name is arbitrary; it could as well be 342, D, or Butternutsquash; a category is just the name for a group of expressions with the same syntactic distribution and semantic type. (Under the Categorical Grammar syntax developed in Chapter 6 it turns out that many category labels are not actually arbitrary since the names themselves encode distributional information. But there will be a set of basic category names, including NP, which remain arbitrary.) Incidentally, there are plurals (like *dogs*) and mass nouns (like *water*), which can function both as nouns (they can occur with the determiner *the* to give an NP) or as NPs as in *Dogs make good pets*, *Water is plentiful on this island*. When they are NPs they have some similarity to proper names: they name “kinds” of objects. (For much more detailed discussion, see Carlson 1980 among others.)

Still, there does seem to be one difference between a name such as *Barack Obama* and an NP like *the president of the US in 2011*. As discussed above, the referent of the latter depends on how the world happens to be. One can imagine worlds in which things had worked out differently and this NP picked out John McCain, or more distant worlds in which this picked out Dennis Kucinich. One can even imagine yet more distant worlds in which it picked out the lead dog of the winning Iditarod team (just imagine a massive change in the political system, a revision in the duties of the president, and a bit of dog worship). But the referent of a proper name like *Barack Obama* is stable; change the world all you want, let in a bit of dog worship and a massive change in the political system, and the referent of this NP remains the same. One can even change some rather fundamental properties—assume that Barack Obama not only is not president but happens to have

been born in Kenya, moved to France, and became a master sommelier. The referent remains the same—just many things about him are changed.²

The account of this proposed by Kripke (1980) is to treat names as *rigid designators*. This simply means that a name does indeed denote a function from worlds to individuals, but the relevant function maps every world to the same individual. Any function which assigns the same value to everything in its domain is called a *constant function*. Note that this hinges on the assumption made earlier: that the domain of individuals is not world-dependent—the set of individuals is independent of the worlds.³ Odd though this may seem at first glance, we can use the tool of counterfactual conditionals to give plausibility both to the claim that the same individual can march around from world to world, and for the claim that proper names are rigid designators. Consider the following sentences:

- (1) If Barack Obama had been born in Kenya, he would not have been eligible to be president of the US.
- (2) If Barack Obama had grown up in France, he would have become a master sommelier.

The semantics in Chapter 2 for counterfactual conditionals leads to the conclusion that there is some individual—call that individual *o*—whose properties can vary from world to world but is nonetheless a single individual. Moreover, the name *Barack Obama* picks out *o* in our world, where he is President of the US in 2013 and was born in Hawaii, and continues to pick

² Notice that we can change large facts about a person—even their birthplace—as we move them from world to world while still considering them the same person. Lest one think that a person's birthplace is a necessary fact about them—rather than a contingent fact—consider the claims of the “Birthers.” These are a group of people who have continually insisted that Barack Obama was actually born in Kenya (and hence ineligible to be president of the US). Notice that while this group insists on a world in which the Barack Obama individual was born somewhere else, they never once disputed that Barack Obama is Barack Obama. The dispute kept the individual constant, and simply centered on a world (not the actual one) in which that same individual had a different birthplace from his actual one (which is Hawaii).

³ One might wonder how could this be, since it is a contingent fact as to who is born, and therefore who is “actual” in any given world. We can, however, assume that there is an equivocation here about the word “existence.” Let us imagine some very abstract notion of “existence” which holds for all individuals who are present in any world. And of course one can also wonder about the proper account of fictional individuals like *Santa Claus*. These worries are all beyond the present scope.

out *o* in the world in (1) in which *o* was born in Kenya, and in the world in (2) in which *o* grew up in France.

While difficult, one can even change more basic facts about the individuals. My dog Mitka happens to be a Siberian husky, but I could meaningfully assert (3a) and even (3b):

- (3) a. If Mitka had been a Labrador Retriever, I would have been able to let him off leash.
b. If Mitka had been a wolf, his fear of thunder would not have let him survive.

Again these sentences make sense if we can take the individual rigidly referred to by *Mitka*, locate him in a different world, and change his properties, while still tracking that it is the same individual. We will thus adopt Kripke's semantics for these. This allows us to account for the observations above, while keeping their semantic type the same as that of other NPs: functions of type $\langle s, e \rangle$. They just happen to be constant functions.

***4.1.** The verb *think* takes as its object (or complement) a sentence as in *Chris thought that the earth was flat*. This verb is a good example of something which crucially combines with *intensions*. The object of Chris's thoughts is not just a truth value 1 or 0, but a set of worlds (all worlds compatible with what he believes). If one wishes to simplify, one can think of this as simply a single world—for the purposes of this question it makes no difference.

Notice that there is an interesting ambiguity in a sentence like the following:

- (i) Bill thinks that the mayor of Providence will win the marinara sauce competition.

See if you can detect what the ambiguity is and describe—perfectly informally—how one might account for the ambiguity. (Do note that this exercise is *not* looking for a formal account, as the tools for that are developed only in Chapter 19. The hope is that the informal intuition might be accessible.) Keep in mind that phrases like *the mayor of Providence* are functions from worlds to individuals; this is crucial. Assume further that in putting the whole semantics together, some actual world is “plugged in” as the argument of the function so that this NP ends up denoting an individual, but the individual can vary from world to world. Incidentally, this phrase is also time-dependent (it actually denotes a

(cont.)

function of type $\langle s, e \rangle$) but this fact can and should be ignored here; keep the time constant.

Now notice that (ii) does not display the same type of ambiguity:

(ii) Bill thought that Buddy Cianci would win the marinara sauce competition.

Comment on why this is not surprising. (A historical note: Buddy Cianci is not the mayor of Providence at the time of writing this book. But he was, and indeed he did make award-winning marinara sauce.)

Before leaving this, we can also note that some NPs fail to refer to anyone in the actual world, giving rise to the classic notion of presupposition failure. The time-honored example (from Russell 1905) is *the (present) King of France*. As mentioned earlier, this book contains no in-depth discussion of presupposition, but we can note that one account is to treat the meaning of such NPs as partial functions from worlds to individuals, undefined for this world.

4.2. Intransitive verbs/verb phrases

The toolbox so far contains as basic building blocks a set of worlds, a set of times, a set of truth values, and a set of individuals. While there is controversy as to whether other primitives (e.g., events) are necessary, one can get surprisingly far using only these sets and constructing other semantic objects using sets and functions built out of these. As a first step, consider the case of sentences with intransitive verbs. First a small bit of housekeeping. Sentences with simple intransitive verbs in English can contain the verbs in the past tense, as in:

- (4) The president coughed.
- (5) Mitka howled.

But we are ignoring the semantics of the past tense (as we wish to ignore times), and so in using examples of this type it should be kept in mind that the semantics is incomplete. Note, incidentally, that we could not give a

fuller account by recasting into the syntactic form which is called the *simple present*, as in (6) and (7), for the semantics of these is even more complex:

(6) The president coughs.

(7) Mitka howls.

Sentences with “eventive” verbs (*coughs*, *runs*, *howls*) in the tense which is called simple present in English actually do not make a statement about an event holding true at the moment of utterance. For example, (6) does not entail that the president is coughing at this moment, but that this happens from time to time with some regularity. These are called *habitual* statements. So we will refrain from using these sentences, and stick to past-tense cases.

We begin with the syntax. We will refer to the category of intransitive verbs as V_I , and will at this point assume that English contains the two phrase structure rules in (8):

(8) a. $S \rightarrow NP \quad VP$ b. $VP \rightarrow V_I$

Of course, a different possibility would be to adopt only the rule in (8a) and simply think of *cough*, *howl*, and so forth as lexical VPs. After all, it was shown above that single words could be lexical NPs, and it would make sense to take the same solution here. But in the present case the situation is more complex, for here transitive verbs (*like*, *kill*, etc.), ditransitive verbs (*give*, *put*, etc.), and intransitive verbs (and many other things that are traditionally called “verbs”) *do* have common properties. They are the class that takes past-tense morphology, the *-ing* suffix, etc. In other words, the morphology treats them as a single class, giving some basis for the traditional notion that these form a single “part of speech.” For now, then, let us stick with tradition and call these all verbs (using a subscript feature as above in order to distinguish their syntactic distribution—and, shortly, their semantic type) which leads us to adopt the two rules in (8). All of this will be modified in Chapter 6 when we turn to Categorical Grammar.

The simplest semantics to associate with the rule (8b) is one which says that meaning of the VP is the same as the meaning of the V_I . Put in the official notation then, (8b) can be recast as follows:

TR-4. If α is an expression of the form $\langle [a], V_I, [[a]] \rangle$, then there is an expression β of the form $\langle [a], VP, [[a]] \rangle$.

The remaining challenge is to determine the type of meaning for the intransitive verb (and hence the VP) itself. This can easily be done without introducing any new primitives into the basic set: let these pick out (in a world and at a time) a *set* of individuals.

At first, this seems odd. But first, note that—as with NPs and Ss—there is a distinction between the *intension* of an intransitive verb and its *extension*. The intension is a function from worlds (and times) to sets; it is only the extension that is a set. Indeed, if you knew everything about the world (and time) that you were living in, and you knew the meaning of the word *cough*, you could determine the set of coughers. So intransitive verbs have meanings of type $\langle s, \langle e, t \rangle \rangle$. To make this a bit more intuitive, we digress briefly to talk about word meaning. Modeling the meaning of intransitive verbs as functions into sets of individuals gives some nice tools for talking about the relationship between the meanings of different words that clearly have certain semantic properties in common.

4.3. A brief look at lexical semantics

Thus while this book is primarily about compositional semantics—how the meanings of smaller expressions (including individual words) combine to give the meanings of larger expressions—that project itself is obviously intimately tied up with *lexical semantics*, i.e., the project of determining the meanings of the basic units (the words). And the tools developed so far are quite useful for lexical semantics as well as compositional semantics. Consider, for example, the fact that (9) entails (10) and that this is clearly a fact about the meanings of *dance* and *move*:

(9) Sabrina danced.

(10) Sabrina moved.

If we stick to the idea that the extension of any intransitive verb is a set of individuals, there is a simple way to express the relation between these two verbs. One part of the meaning of $[[\text{dance}]]$ is that for every individual in the set it picks out, that individual is also in the $[[\text{move}]]$ set. Put differently, the extension of $[[\text{dance}]]$ at any world is a subset of $[[\text{move}]]$ (in that world). (Of course $[[\text{dance}]]$ is richer than just this, so the full lexical entry will say more.)

This also provides tools for the meanings of certain words which are often thought of as being decomposable into other bits of meanings. To illustrate, we turn from intransitive verbs to nouns. As will be discussed more fully in Chapter 8, nouns also plausibly have as their meanings world- and time-dependent sets of individuals. Some noun meanings lend themselves to a combination of other concepts. A famous example (see, e.g., Katz and Fodor 1963) is *bachelor*, which in early work on lexical semantics within generative grammar was treated as being composed of a few primitive concepts. Oversimplifying somewhat, we can reconstruct this idea by saying that $[[\text{bachelor}]]$ is a function from worlds and times to sets of individuals such that for any world w and time i $[[\text{bachelor}]]$ at w and i is the set: $\{x | x \in [[\text{male}]] \text{ (at } i \text{ and in } w) \text{ and } x \in [[\text{adult}]] \text{ (at } i \text{ and } w) \text{ and } x \notin [[\text{married}]] \text{ (at } i \text{ and } w)\}$. This recapitulates—in model-theoretic terms—the early “lexical decomposition” analysis of this in terms of primitive concepts. Of course the above needs refining, but we have the tools to make the necessary refinements. For a bachelor is not any unmarried male adult, but one who has never been married. Thus we can revise the above by requiring that for each x in the $[[\text{bachelor}]]$ set at world w and time i , there is no time i' prior to and including i at which $x \in [[\text{married}]]$.

4.2. The situation is even more interesting: possible worlds also appear to be necessary to give a full treatment of even a simple word like *bachelor* which is often trotted out in lexical semantics courses to illustrate how words decompose into other concepts. Thus most speakers have the intuition that a Catholic priest is not felicitously referred to as a *bachelor*, nor is a gay man in a state which does not recognize same-sex marriage.

- (i) How can the use of possible worlds provide a way to capture these intuitions?
- (ii) Having done (i), does this also us to eliminate any other part of the above definition? Note too that we did not put $[[\text{human}]]$ in the definition above although this might have struck the reader as a missing ingredient. But given (i), do we need it?

Moral: A speaker’s ability to use perfectly ordinary words like *bachelor* involves counterfactual reasoning.

4.4. Back to the compositional semantics

With this much, we can now formulate the semantics for the rule in (8a). Recasting into the official notation yields TR-5 (given extensionally):

TR-5. If α is an expression of the form $\langle [a], \text{NP}, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], \text{VP}, [[\beta]] \rangle$, then there is an expression γ of the form $\langle [\alpha-\beta], \text{S}, 1 \text{ if } [[\alpha]] \in [[\beta]], \text{ and } 0 \text{ otherwise} \rangle$.

Notice that TR-5 as formulated above leaves no room for treating certain kinds of presupposition failure as a case of a sentence being neither true nor false. In other words, for the sake of discussion take *stop-smoking* as a single intransitive verb. It is sometimes assumed that when predicated of someone who never smoked the result is neither 1 nor 0 but is simply undefined. The semantics above provides no way to accommodate for this view; every sentence of the form *NP stopped smoking* will be either true or false. We remedy this directly below.

4.3. Give the intensional version of TR-5.

TR-5 is based on treating the meaning of an intransitive verb as (relative to a world and time) a set of individuals. But as shown in Chapter 2, any set can be recast as the characteristic function of that set. Thus an intransitive verb like *cough* or *howl* can instead be taken as picking out (in any world and time) a function from individuals to truth values. Using e to mean the set of all individuals, this then picks out a function of type $\langle e, t \rangle$ (so its full meaning is a function of type $\langle s, \langle e, t \rangle \rangle$). Using functions rather than sets has two potential advantages. First, it provides a more general tool for modeling meanings, as will be clear later. Second, it is a richer notion than the set version because potentially there could be *partial functions* (in this case, a function defined only for a subset of the domain of individuals). If these do exist, one can always recover a set of individuals from a (possibly partial) function of type $\langle e, t \rangle$: the relevant set is the set mapped to true by the function. But one cannot recover the function from the set (if something is not in the set, it could be that it is assigned 0 by the function, or it could be that it is simply not in the domain of the function). For the most part this book will make little use of partial functions, but will leave open the possibility of using it for presupposition failure. Under this technique

$[[\text{stop-smoking}]]$ maps an individual to true (at a time i) if that individual smoked at a time earlier than i and does not smoke at i ; false (at time i) if that individual smoked at a time earlier than i and also smokes at i , and undefined otherwise.

Having made this revision on the semantic types, we revise TR-5 as follows (given extensionally here):

TR-5'. If there is an expression α of the form $\langle [a], \text{NP}, [[a]] \rangle$ and an expression β of the form $\langle [\beta], \text{VP}, [[\beta]] \rangle$, then there is an expression of the form $\langle [\alpha-\beta], \text{S}, [[\beta]] \rangle$ ($[[\alpha]]$).

If there are partial functions and if $[[\alpha]]$ is not in the domain of $[[\beta]]$ then we have undefinedness (and hence the result is undefined at the world and time in question); otherwise the VP-function applies to the individual in question and yields either 1 or 0.

4.4. Give TR-5' in its intensional form.

4.5. Illustrating the syntactic/semantic composition

Let us illustrate the workings of the fragment that has been developed so far (along with a suitable lexicon). Consider (11).

(11) The disobedient husky escaped and the obedient husky slept.

We assume the lexical items *escaped*, *slept*, etc. of category V_1 with meanings of type $\langle s, \langle e, t \rangle \rangle$; and *the-disobedient-husky* and *the-obedient-husky* of category NP with meanings of type $\langle s, e \rangle$. Extensionally, a V_1 denotes a function in $\langle e, t \rangle$ and an NP denotes some member of e . If we actually knew all the facts about the world in question, we could show out the actual extensions of each of these at that world. By way of illustration, imagine a “universe” with four individuals $\{a, b, c, d\}$. Further, imagine a world w_1 , whereby $[[\text{the-disobedient-husky}]](w_1) = a$ and $[[\text{the-obedient-husky}]](w_1) = c$. Assume further that $[[\text{escaped}]](w_1)$ is the function mapping a and c to 1 and b and d to 0 (thus it characterizes the set $\{a, c\}$, while $[[\text{slept}]](w_1)$ maps b to 1 and a, c , and d to 0 (thus it characterizes the set $\{b\}$). We can now use the notation in (12) as one way to illustrate how it is that the grammar proves (11) as well-formed and puts together its meaning.

In the semantic part, we are showing *only* the extension of each expression at w_1 . We give each expression as a triple of phonology, category, meaning (actually, extension). For visual ease we will omit the brackets $\langle \dots \rangle$ around each triple, and we will omit the brackets $[\dots]$ and $[[\dots]]$ around the sound and meaning parts respectively.

- (12) the-disobedient-husky; NP; a
 escaped; V_I ; $a \rightarrow 1, b \rightarrow 0, c \rightarrow 1, d \rightarrow 0$
 these two lines just show the information that is in the lexicon
 escaped; VP; $a \rightarrow 1, b \rightarrow 0, c \rightarrow 1, d \rightarrow 0$ (by TR-4)
 the-disobedient-husky escaped; S; 1 (by TR-5')
 the-obedient-husky; NP; c
 slept; V_I ; $a \rightarrow 0, b \rightarrow 1, c \rightarrow 0, d \rightarrow 0$ (these two lines again given by the
 lexicon)
 slept; VP; $a \rightarrow 0, b \rightarrow 1, c \rightarrow 0, d \rightarrow 0$ (by TR-4)
 the-obedient-husky slept; S; 0 (by TR-5')
 the-disobedient-husky escaped and the-obedient-husky slept; S; 0 (by TR-1)

There are many other ways one could illustrate this. Since we generally do not have all the facts of some actual world at our disposal, we could write the semantic composition more generally. For example, the semantic part of the fourth line can be rewritten as follows: $[[\text{escape}]]^w ([[\text{the-disobedient-husky}]])^w$. Remember that formulas like this contain a hidden clause at the beginning: so this says that for all worlds w , the value of this expression is the value $[[\text{escaped}]]$ at w applied to the value of $[[\text{the-disobedient-husky}]]$ at w . Thus one can show part of the composition as in (13):

- (13) the-disobedient-husky; NP; $[[\text{the-disobedient-husky}]]$ w
 escaped; V_I ; $[[\text{escaped}]]^w$
 escaped; VP; $[[\text{escaped}]]^w$ (by TR-4)
 the-disobedient-husky escaped; S; $[[\text{escaped}]]^w ([[\text{the-disobedient-husky}]])^w$ (by
 INT-TR-5')

In fact, in general we will illustrate the semantics this way (since we could not list the actual function which corresponds to, e.g. $[[\text{escaped}]]$). Among other difficulties in doing that, we would have to list this out for every possible world. But it is sometimes useful to pick a world and demonstrate the composition extensionally as in (12) as a reminder that the semantics is not computing formulas, but actual model-theoretic objects. Formulas—as in (13)—are just what we humans need in order to write out those objects. It is also generally not necessary to put in each rule that licenses the next line;

this is useful only when it serves some clarificational purpose. The literature contains other methods for visually displaying the syntax and semantics at work. Often this is shown as an annotated tree. A traditional tree shows just the phonological part—what precedes what—and the categories of each expression, but not the meanings). One could then “decorate” each node label with the semantics.

4.5. Consider the following model of a tiny universe. We will be concerned with just one world and one time, so we can consider everything extensionally. Our universe has four individuals $\{a, b, c, d\}$. At the world in question, $[[\text{howl}]]$ maps a and c to 1, b and d to 0. $[[\text{cough}]]$ maps b to 1 and all other individuals to 0. $[[\text{Balto}]]$ picks out a ; $[[\text{the hungry dog}]]$ picks out d . Show the full syntactic and semantic composition for

(i) Balto howled and the hungry dog coughed.

In this case, the task is to show the actual extension of each expression at the world in question; thus use (12) as the model for showing this.

4.6. To make things interesting, and to give a flavor for intensionality, let us add in another world. World 1 (which we will label as w_1) will be exactly the world set up in Exercise 4.5. World 2 (labeled w_2) is as follows: $[[\text{Balto}]]$ still picks out a (of course, since it is a rigid designator); $[[\text{the hungry dog}]]$ picks out c ; $[[\text{is-howling}]]$ maps all four individuals to 0; and $[[\text{is-coughing}]]$ maps a , c , and d to 1 and b to 0. Now show the syntactic and semantic composition where at each step you should show the full function from each of the two worlds to its value, and show how this is computed step by step.

5

Transitive verbs: Resolving an apparent syntax/semantics mismatch

5.1. First pass: Straw Man's solution	67	5.4. Back to the drawing board: Curry'ed functions	76
5.2. Abandoning Direct Compositionality	70	*5.5. A note on three-place verbs	84
5.3. Hidden S conjunction?	73		

This chapter is somewhat didactic—for a reason. The immediate goal is to discover the semantic type of transitive verbs, which could be done in a few pages. But we will instead go down a lengthy and winding path (which will turn out to be fruitless). This involves setting up a somewhat naive “Straw Man,” whose hypothesis requires giving up Direct Compositionality and then leads to greater and greater complications. After Straw Man has taken enough twists and turns, we abandon that path, and turn to a simple Direct Compositional solution. In one sense the lengths to which Straw Man goes is a bit silly, for in reality most work in modern semantics does not take the Straw Man position. The ultimate solution arrived at in this chapter is fairly standard (even in non-Direct Compositional theories).¹ But this is precisely what makes this domain an important one. For here the Direct Compositional solution turns out to be

¹ An exception to this is work done under the rubric of *Neo-Davidsonian event semantics*; see, e.g., Parsons (1990).

reasonably obvious with some fairly simple tools that allow the semantics to be set up in such a way as to respect the syntax. Perhaps, then, this basic strategy extends to other domains; perhaps slightly more subtle tools can be pressed into service in other cases that appear to challenge Direct Compositionality.

5.1. First pass: Straw Man's solution

Consider sentences with transitive verbs like those in (1):

- (1) a. Juliet loves Romeo.
- b. The pig chased the wolf.
- c. Lee kissed Sandy.

(Once again the eventive verbs like *chase* and *kiss* are put into the past tense to avoid the habitual reading that emerges with present-tense morphology.) What kind of meaning should be assigned to *kiss*, which is often thought of as a relation between individuals? Straw Man consults an elementary logic book, in which sentences like (1c) are translated into a formula of first-order predicate logic and are represented as in (2):²

- (2) $K(l,s)$

Recall that an intransitive verb has as its value a (world/time-dependent) function of type $\langle e,t \rangle$. If asked how to construe (2) as a way to name a model-theoretic object, a reasonable answer is to take l and s in the formula above to denote individuals, and to take $[[K]]$ to be a function whose domain is the set of all ordered pairs of individuals and whose co-domain is the set of truth values. In other words, (l,s) picks out an ordered pair of two individuals, and $[[K]]$ is a function mapping such a pair to true or false. In set terms, $[[K]]$ characterizes a *set of ordered pairs of individuals*.

Straw Man applies this to the semantic composition of English, and concludes that English $[[kiss]]$ is like $[[K]]$: a function from the set of ordered pairs of individuals to t . (Recall that the set of all ordered pairs whose first member is in set A and second is in set B is represented as $A \times B$.) Thus the

² While we will show that this is not a good way to represent the semantic composition of English, the point here is *not* to criticize the use of such formulas in a logic textbook: most work using formulae like this does not claim to be modeling the grammar of natural language. The goal of the logician is quite different (to provide a way to represent valid reasoning systems).

type of meaning for a transitive verb would be $\langle e \times e, t \rangle$ (or, more accurately, $\langle s, \langle e \times e, t \rangle \rangle$). Incidentally, one need not have taken a logic class to come up with the hypothesis that transitive verbs take ordered pairs as their arguments. We often speak of *kiss* as denoting a (two-place) relation between two individuals. But recall that the formal notion of a (two-place) relation is a set of ordered pairs (and of course any set can be recast as the characteristic function of that set, which gives the type above).³ It is also common parlance to say that the verb applies to a pair of a “kisser” and “kissee,” or to an “agent” and a “theme,” and this also leads naturally to the hypothesis that the type for transitive verbs is as above. There is, in fact, nothing wrong with this informal terminology; the central question of this chapter is whether the type above is the right way to formally model the informal intuition.

Let us call the syntactic category of a transitive verb V_2 . Then the semantic type assumed above leads us to posit FR-1. (Fictional rules that will quickly be discarded will be notated as FR.) To minimize notational collision, an ordered pair whose first member is a and second is b will (non-standardly) be notated as $\{a, b\}$ (with boldface curly brackets to distinguish from ordinary curly brackets which are used for sets). This notation is not ideal, but unfortunately all of the other types of bracket are used elsewhere in our formalism (the standard representation of an ordered pair is (a, b) , but parentheses are also used to indicate the argument of a function). FR-1 is given extensionally here; the interested reader can supply the intensional version:

FR-1 If there is an expression α of the form $\langle [a], NP, [[a]] \rangle$, an expression β of the form $\langle [\beta], NP, [[\beta]] \rangle$ and an expression γ of the form $\langle [\gamma], V_T, [[\gamma]] \rangle$ then there is an expression δ of the form $\langle [a-\gamma-\beta], S, [[\gamma]](\{[[a]], [[\beta]]\}) \rangle$.

Since the meaning of the transitive verb combines with the meanings of its subject and object simultaneously in the semantics (i.e., by applying to an ordered pair) the syntax also directly combines the two NPs with the transitive verb.

But wait, says the syntactician. We learn from our introductory syntax texts that this cannot be the right syntax. The phrase structure rule embodied in FR-1 is $S \rightarrow NP V_2 NP$, but we know (or at least we think) that *loves Juliet* in (1a) is a constituent and similarly for *kissed Sandy* in (1c). Moreover, we know (or at least we think) that these expressions are of exactly

³ A two-place relation means a set of ordered pairs. There are also three-place relations which are sets of ordered triples, and more generally one can speak of ordered n -tuples.

the same category as expressions consisting of only single intransitive verbs—i.e., they are VPs. It is actually worth occasionally reminding oneself of the arguments that lead to this conclusion; we mention just one here. Expressions like *kissed Sandy*, *watered the plants*, or *fed the pig* can conjoin with each other, and can also conjoin with simple intransitive verbs:⁴

- (3) a. Lee watered the plants and fed the pig.
 b. Lee watered the plants and meditated.
 c. Lee meditated and fed the pig.

If transitive verbs were introduced in the syntax in the flat way given in FR-1, we would need three additional syntactic rules for these:

- (4) a. $S \rightarrow NP V_T NP$ and $V_T NP$
 b. $S \rightarrow NP V_T NP$ and V_I
 c. $S \rightarrow NP V_I$ and $V_T NP$

But this would still not be enough for these conjunctions can be iterated:

- (5) a. Lee fed the pig and chased the wolf and meditated.
 b. Lee fed the pig and meditated and watered the plants and studied.

Additional rules would thus be needed, but since one can iterate these indefinitely, no amount of new rules will be enough. Positing a category VP solves all of this: simply add a syntactic rule of the form in (6), and assume the two VP rules in (7):

- (6) $VP \rightarrow VP$ and VP
 (7) a. $VP \rightarrow V_I$
 b. $VP \rightarrow V_T NP$

Note that the remarks above concerning *and* extend also to *or* as in (8), and so we also add the rule in (9):

- (8) a. Lee milked the cow or fed the pig.
 b. Lee milked the cow or meditated.
 (9) $VP \rightarrow VP$ or VP

⁴ There are actually many versions of generative grammar—including earliest Transformational Grammar—in which it is assumed that all verbs must have a subject (at least at some level) in the syntax. This would invalidate the argument below. For *fed the pig* in (3a) for example would at some level have a deleted or silent subject and be a full sentence; and all conjunction would actually involve S conjunction. We will not consider these here, but see the discussion in section 5.3.

Ultimately, there are many refinements one could make to (6) and (9), but these formulations are fine for now as we turn back to the semantics.

All is well as far as the syntax is concerned, but since the goal is a theory of syntax *and* semantics it is striking to notice that the rules above give only the syntactic side. What would be the semantics that goes along with (7b)? Well, if we continue with the Straw Man hypothesis (i.e., $[[\text{kiss}]]$ is a function that takes as its argument some ordered pair) there can be none. The reason is simple: V_2 is not of the right type to combine with just the one individual denoted by the object NP. And since there is no semantics for (7b), it also does not make sense to ask about the semantics associated with (6) and (9); there is none.

5.2. Abandoning Direct Compositionality

Straw Man is stubborn and does not wish to give up on the idea that $[[\text{kiss}]]$ is a function whose domain is $e \times e$. But he has learned his syntax well and is happy enough with the syntactic rules above. Thus he decides to abandon the hypothesis of Direct Compositionality. He will leave the syntax of (7) alone, and adopt a different model of grammar from the one we have been assuming so far. Before developing this, a word about the role of trees. Probably a bit surprisingly for a linguist, there has so far not been a single tree drawn in this book. This is because we have simply stated the rules, and the rules so far make no reference to trees. Consider a complex sentence like, for example, *Mitka howled and Porky grunted*. (Porky made his debut in semantics in Lewis 1970.) One can of course draw a tree for this, but the tree is just a representation of the steps in a proof of the well-formedness of the sentence. Suppose that the rules combining expressions do nothing more than put such expressions next to each other: this is known as *concatenation* (this assumption will be subject to greater scrutiny in 5.5). Put differently, the syntactic portion of the rules are just those that could be written as a context-free phrase structure grammar (see Chapter 3, n. 4).⁵ Under this

⁵ It is well known that ultimately a theory of grammar will need to include rules beyond just context-free phrase structure rules (which only concatenate expressions). While the jury may remain out for the case of English, considerable debate about this point during the 1980s (especially within the theory of Generalized Phrase Structure Grammar) showed that additional devices beyond context-free phrase structure grammars will be needed for other languages. But just how much “extra” is needed remains open. See section 5.5 for discussion of these points.

view, the grammar never needs to refer to trees as it never refers to the internal structure of some expression. A tree is always just a way to represent the proof of the well-formedness of some expression, and given the Direct Compositional hypothesis, it is also a representation of the steps in the semantic composition. Under Direct Compositionality, then, combined with a rather impoverished view of the set of syntactic operations, a tree is a convenient representation for the linguist but it is not something that the grammar itself “sees.”

But of course much work in syntax and semantics assumes that the grammar does have access to the trees, and the view of semantics to which we are about to turn also makes this assumption. Thus in order for Straw Man to both adopt (7b) in the syntax and at the same time maintain that $[[\text{kiss}]]$ is a function of type $\langle \text{exe}, t \rangle$ (or the intensional counterpart), the syntax and semantics must be pulled apart. The syntax not only proves strings well-formed but keeps track of the steps used so that in the end it is a proof of the well-formedness of a tree. Moreover, it “feeds” into the semantics as a separate system: a system which provides interpretations for trees. Hence we adopt the syntactic rules in (7), and add a semantic rule interpreting a tree as in FR-2. (The notation here is inspired by the notation developed in Heim and Kratzer 1998, although they do not use it for this case as they adopt the same solution for transitive verbs as is developed later in this chapter.)

$$\text{FR-2} \quad \left\| \begin{array}{c} \text{S} \\ \swarrow \quad \searrow \\ \text{NP}_1 \quad \text{VP} \\ \quad \swarrow \quad \searrow \\ \quad \text{V}_T \quad \text{NP}_2 \end{array} \right\|$$

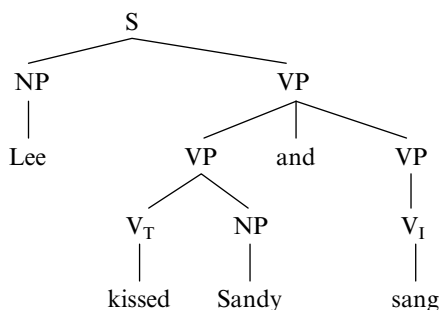
$$= \quad [[V_T]] \left(\{ [[NP_1]], [[NP_2]] \} \right)$$

(This is given extensionally; it is easy enough to fold in intensionality.) If this move is made, the entire fragment would be revised; each syntactic/semantic rule pair developed up to this point will be stripped of its semantic part. However, in all of the cases we have discussed so far the necessary revisions are straightforward; the revisions are left to the reader in the following exercise.

5.1. Take the rules in the fragment so far (this means only those rules labeled TR). For each one, assume that they give just the syntactic and phonological part (they could be stated as phrase structure rules or they could be stated in the notation used here). Assume further that for each such rule there is a separate semantic rule whose input is a tree. Then state each of these semantic rules, using FR-2 as a model for the statement of the semantic rules. You should note that FR-2 differs from the other rules you will be constructing in that all of those rules have as their input a *local tree*—that is, only a mother node and one or more daughter nodes—while FR-2 (by necessity) refers to a bigger chunk of tree—mother, daughter, and granddaughters.

The Direct Compositional strategy vs “syntax feeds semantics” strategy is discussed in Chapter 7, but for now we can note that the syntax feeds semantics strategy for this particular case comes at a cost. In the case of a sentence like *Lee kissed Sandy*, the fact that *kissed Sandy* has no meaning was remedied by the adoption of FR-2. But then, how does the semantics work for the case of conjoined VPs (i.e., cases where VPs are connected by *and* or *or*)? Recall that Straw Man posits the syntactic conjunction rule in (7). Now consider the syntactic structure in (10):

(10)

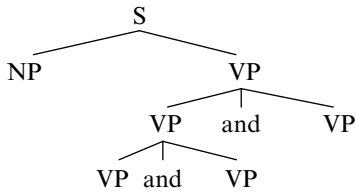


Do the rules in Straw Man’s grammar provide an interpretation for this sentence? The answer is no: there is no rule to interpret either the topmost VP here nor the entire S. FR-2 is not relevant; it interprets only Ss of the form $[_S \text{ NP } [_{VP} \text{ V}_T \text{ NP}]]$, and none of the rules that you will have formulated in your answer to Exercise 5.1 will help either. So there would need to be an additional rule to interpret this tree.

5.2. Formulate the relevant rule (remembering to keep in mind the types that Straw Man is assuming).

But of course even adding the rule asked for in Exercise 5.2 does not complete the task. For the additional rules for (10) will do nothing for cases like those in (5), which have (as one possibility) the shape in (11):

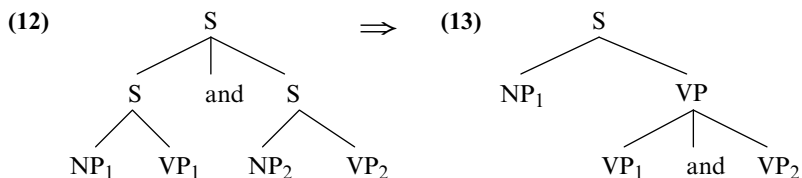
(11)



Due to the recursive nature of the *and*-rule, Straw Man could never finish stating the semantic rules. This is completely parallel to the reasoning as to why the syntax needs a VP constituent: the same reasoning leads to the conclusion that we need to find a meaning for each VP, which will be done in section 5.4.

5.3. Hidden S conjunction?

But first, Straw Man makes one final attempt to save his hypothesis. And this is to posit that the semantics actually does not interpret structures with VP conjunction. Rather, at the level at which these are interpreted, these are actually the conjunction of two sentences. This is actually not a straw man; this view has been taken in one form or another in many theories of grammar, and has been implemented in various ways. For example, in early Transformational Grammar it was assumed that there were no phrase structure rules like that in (6) and (9). Sentences like (3) were actually at an underlying level (deep structure) instances of S conjunction, with a transformational rule called Conjunction Reduction which mapped trees of the form in (12) to those in (13), provided that NP₁ was identical to NP₂ (see, e.g., Jacobs and Rosenbaum 1968):



(The rule can be generalized to extend to the case of *or*.) Additionally, it was assumed that the input to the semantic interpretation was the deep structure, and so the semantics would be assigning a meaning to structures of the sort shown in (12) and not to (13). Since we already have semantics for S conjunction, there would be no need to worry about interpretation for the case of VP conjunction. Hence for a sentence like *Romeo kissed Lee and watered the plants* we don't need to worry about giving a meaning for *kissed Lee*. The semantics would separately interpret the two sentences *Romeo kissed Lee* and *Romeo watered the plants* (along with the rule interpreting the conjunction of the two). FR-2 will do the job. (One might notice that under this view of conjoined sentences, the argument given earlier for a VP constituent has been undermined, but we will not pursue the consequences of this.) A variant on this strategy is to view the picture in reverse. That is, assume that the phrase structure rules do give something like (13) directly, but this is mapped to (12) (as its "Logical Form"). The semantics operates so as to compositionally interpret that level. Again FR-2 (combined with the other rules suitably reformulated in tree terms, as in Exercise 5.1) will now be adequate.

But the hidden S conjunction strategy also comes at a cost. First and foremost the syntax has again been complicated, no matter which variant is chosen. Either variant posits an additional level of representation and hence needs a set of explicit rules to map one level to the other. There is also an empirical difficulty centering on sentences whose subjects contain quantifiers. Consider the pairs in (14) and (15). In each pair the first involves conjunction or disjunction of just bare VPs, and the second is the corresponding conjoined Ss with (syntactically) identical subjects:

- (14) a. Some pig chased the wolf and grunted.
 b. Some pig chased the wolf and some pig grunted.
- (15) a. Every pig chased the wolf or grunted.
 b. Every pig chased the wolf or every pig grunted.

The meaning of (14b) is weaker than (14a); in the former the chaser and grunter must be the same. In (14b) the chaser and grunter could happen to

be the same, but they need not be.⁶ Hence (14a) entails (14b) but not vice versa. The reverse situation holds in the *every* case. Here (15a) is weaker—the most salient reading for (15a) is each pig did one or the other of these things, but they need not have all been in agreement as to which act to perform. (15b) has the stronger requirement that they all did the same thing. Incidentally, it is true that (15a) also (at least marginally) has a second reading which is the same as (15b); this is best brought out by the continuation in (16):

(16) Every pig chased the wolf or grunted—but I can't remember which.

Nonetheless, the most salient reading of (15a) is clearly the weaker one and is true in situations in which (15b) is not. (The second reading is accounted for in Chapter 11; its existence does not affect the point at hand here.)

Consider a theory in which the Logical Form in (12) is related to the surface form in (13) and where the requirement for this is that the two NP subjects in the two clauses in (12) must be identical. Then, of course, we get the wrong prediction. In the cases above, the (a) sentences should have the same meaning as the (b) sentences. As noted above, there is a secondary reading of (15a) which is synonymous with (15b) but this is of little consolation since the correct theory needs to also pair (15a) with its primary meaning. To be sure, there are ways to solve these problems with more complex Logical Forms and additional conditions on the rules effecting the mapping to (or from) Logical Forms.⁷ But the methods involved require yet further rules. Perhaps it is time to abandon Straw Man's path.

⁶ It might appear at first glance that (14b) requires the wolf-chasing pig and the grunting pig to be different pigs. But closer inspection reveals that this is just an *implicature* (see section 3.2) and not part of the truth conditions. If a speaker knew that a single pig did both, they would more likely have used (14a)—hence the conclusion on hearing (14b) that different pigs are involved. But this can't really be part of the truth conditions. For if I hear a grunting pig and have no idea who it is and at the same time have some other indirect evidence about there being a wolf-chasing pig, I can certainly say (14b) with no commitment one way or the other as to whether it is the same pig. (And if it turns out to be the same pig, I have not said anything false.)

⁷ Readers familiar with the “Quantifier Raising” solution to quantified NPs (see section 14.2) might think that one automatically gets the right meanings of the (b) sentences once one combines Quantifier Raising with the rule that produces the two-sentence LF. But it is not automatic, for a principle is needed to ensure that the two processes happen in the right order. We leave it to the interested reader to verify this.

5.4. Back to the drawing board: Curry'ed functions

5.4.1. Recasting the meaning of transitive verbs

The complications above are all the result of Straw Man's insistence that the meaning of a transitive verb is a function from ordered pairs to truth values. The problem is that this makes semantic composition look very different from the syntax, for the semantics treats the composition as "flat" while the syntax does not. But there is a simple way to recast things so that the semantics respects the syntax. The technique to be used here relies on the observation that any function whose domain is a set of ordered n -tuples (in this case, ordered pairs) can be mapped to a function that takes the arguments one at a time. The observation that this can be done for any such function is due originally to Schönfinkel (1924) and later H. B. Curry (see, e.g., Curry and Feys 1958). Although Schönfinkel's work is earlier, this is generally known as *Currying* a function (although see Heim and Kratzer 1998 who do use the historically more accurate term *Schönfinkelization*).

In the case at hand, we begin with the original $[[K]]$ (a function from $e \times e$ to t) and recast this as a function which takes one individual and returns a function of type $\langle e, t \rangle$. This can be done without loss of information—this is just repackaging the meaning. Thus $[[kiss]]$ is a function of type $\langle e, \langle e, t \rangle \rangle$. Note that there are actually two different ways that one can recast the initial logician's $[[K]]$ (which maps an ordered pair $\{x, y\}$ to true just in case in the world in question x does kiss y). One way is to create a new function such that for any x and y such that $[[K]](\{x, y\})$ is true, $[[kissed]](x)(y)$ is true. In other words, the "kisser" is the first argument of the new function, and the "kissee" is the second. The other mapping does just the reverse: for any x and y such that $[[K]](\{x, y\})$ is true, then $[[kissed]](y)(x)$ is true.

It turns out that the second strategy is the correct one, given the assumption that meanings are packaged in such a way as to reflect the syntactic composition. Our new function combines with the meaning of the *syntactic direct object* first, because this is what happens in the syntax. It combines next with the subject. This is often confusing when first encountered because of the left-to-right order in which we write things: in a sentence like *Lee kissed Sandy*, the semantic composition will be $[[kissed]] ([[Sandy]]) ([[Lee]])$. (The other meaning can be thought of as the meaning of *was-kissed-by*).

To illustrate in greater detail, suppose that the universe consists of three individuals, *a*, *b*, and *c*. We will, moreover, be considering just one world w_1 . To use English to describe the facts of the world, let it be the case that *a* kisses *b* and *c*, *b* kisses *c*, and *c* kisses *a* and *c*. Then the function $[[\text{kiss}]]$ in w_1 is as follows:

$$(17) \quad \begin{aligned} a &\rightarrow \begin{cases} a \rightarrow 0 \\ b \rightarrow 0 \\ c \rightarrow 1 \end{cases} \\ b &\rightarrow \begin{cases} a \rightarrow 1 \\ b \rightarrow 0 \\ c \rightarrow 0 \end{cases} \\ c &\rightarrow \begin{cases} a \rightarrow 1 \\ b \rightarrow 1 \\ c \rightarrow 1 \end{cases} \end{aligned}$$

In order to get any mileage out of this, we also need to make explicit what was implicit in the discussion above—i.e., the syntactic and semantic combinatory rule for combining transitive verbs with their objects:

TR-6. If α is an expression of the form $\langle [\alpha], V_T, [[\alpha]] \rangle$ and β is an expression of the form $\langle [\beta], NP, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [\alpha-\beta], VP, [[\alpha]]([[\beta]]) \rangle$.

Hence consider the composition of *Lee kissed Sandy*. Assume that $[[\text{Lee}]]$ (in all worlds) is *a* and $[[\text{Sandy}]]$ is *b*. Then *kissed Sandy* has a meaning; it is the function shown in (17) applied to the argument *b*, so it is that function of type $\langle e, t \rangle$ which maps *a* to 1 and *b* and *c* to 0. In set terms, *kissed Sandy* characterizes the set of Sandy-kissers (which in this case is the singleton set $\{a\}$). Then this VP combines with its subject by the syntactic and semantic rule TR-5', and the resulting semantics is 1, as we would expect.

5.3. The picture above in (17) is the extension of $[[\text{kiss}]]$ in a world that we have labeled w_1 . Now take a different world w_2 in which the facts are that *a* kisses *a*, *b*, and *c*, *b* kisses no one, and *c* kisses only *c*. (Ignore the obvious lack of pragmatic plausibility here!)

(cont.)

First, show the extension of $[[\text{kiss}]]$ at w_2 where this again will be shown as a Curry'ed function of the same basic structure as (17). Then show the full syntactic and semantic composition of *Sandy kissed Lee*, where at each step you will be showing the meaning of each expression as a function from worlds to some other object. (You already have $[[\text{kiss}]]$; it is a function mapping w_1 to the object shown above in (17) and mapping w_2 to the new object that you will have constructed.) The two NPs are listed in the lexicon and each is a constant function from worlds to individuals as given above. So your task is to show how the syntax and the semantics put this all together step by step. Do this using (12) in Chapter 4 as the model.

Does this strategy introduce undue complexity into the modeling of meanings? Not at all. We have simply taken $[[\text{kiss}]]$ to be a function whose *co-domain* itself is a function. The general approach of the kind of model-theoretic semantics being assumed here is to have a small set of primitive model-theoretic objects (worlds, times, truth values, and individuals) and take other kinds of meanings to be constructed from these using the notion of functions.⁸ There is no reason to be surprised to find that certain expressions denote functions which themselves have functions as their co-domain (or, for that matter, as their domain; see Chapter 10). In fact, such objects are already in the system. In the fully intensional semantics, the meaning of an intransitive verb, for example, is a function of type $\langle s, \langle e, t \rangle \rangle$.

Incidentally, one finds different notations for the semantic composition of a sentence like *Lee kissed Sandy*, even among researchers who agree that $[[\text{kiss}]]$ is the Curry'ed function of type $\langle e, \langle e, t \rangle \rangle$ (combining first with the individual denoted by the object). Given the world above, the composition is most accurately represented as $[[\text{kiss}]](b)(a)$. However, one sometimes finds more reader-friendly notation when the actual step-by-step composition doesn't matter, and so one might see something like $[[\text{kiss}]](a, b)$, which looks like it is going back to the hypothesis that $[[\text{kiss}]]$ takes

⁸ There are debates as to whether others are needed—see especially the work in event semantics (e.g., Parsons 1990) which takes events as primitives. But these will suffice for what we will be considering in this book.

an ordered pair as argument. But generally this is intended simply as a way to make things easier to read; this is called the relational notation. It's ultimately helpful to get used to going back and forth between the two notations.

*5.4.2. Currying more generally

The above illustrates the technique of Currying to a function that has as its domain a set of ordered pairs, but this is quite general. For any function f which takes an ordered n -tuple as its argument to give some result x , there is a corresponding function f' which takes a single argument, and returns as value a function g from an $n-1$ tuple to x . This can apply recursively, until we have a function that takes its arguments one at a time.

Generalizing the remarks above, concerning the fact that there are different ways to Curry a function like $[[K]]$, there are many ways to effect the mapping from f to the one-place function f' , depending on which position of the n -tuple is peeled off (A one-place function is one which takes just a single argument, not an ordered tuple). Call f'^{-i} the function that peels off the i th position of the n -tuple. Then there is a unique way to map f to f'^{-i} (and vice versa). Notice that the relationship between the naive meaning of *kiss* (a function taking ordered pairs of kissers and kissees) and our hypothesized actual meaning of English *kiss* is such that $[[kiss]]$ is $[[K^{-2}]]$ since we took the second member of the ordered pair to be the first argument of $[[kiss]]$. This is crucial: this was done to model the way in which the syntax works, as the syntax first combines *kiss* with its object and then with its subject (and, simply because it is traditional, we took “naive-kiss” to map a pair $\{a,b\}$ to true just in case a kisses b).

***5.4.** Take a set A with two members $\{a,b\}$, a set B with two members $\{c,d\}$, a set C with two members $\{e,f\}$, and a set D with three members $\{1,2,3\}$. Consider a function f whose domain is the set of ordered triples in $A \times B \times C$ and whose co-domain is D . (In other words, it takes each triple that can be composed by taking a member of A , a member of B , and a member of C and maps each such triple into some member of D .) The actual function is as follows:

(cont.)

$\{a,c,e\} \rightarrow 1$
 $\{a,c,f\} \rightarrow 1$
 $\{a,d,e\} \rightarrow 3$
 $\{a,d,f\} \rightarrow 2$
 $\{b,c,e\} \rightarrow 1$
 $\{b,c,f\} \rightarrow 3$
 $\{b,d,e\} \rightarrow 2$
 $\{b,d,f\} \rightarrow 3$

Convert this to the corresponding Curry'ed function of type $\langle C, \langle B, \langle A, D \rangle \rangle \rangle$.

5.4.3. Solving Straw Man's problems

Let us consider whether this strategy solves the problems encountered by Straw Man. First, as seen above, a VP such as *kissed Sandy* now has a meaning. And, having Curry'ed the transitive verb, this VP has a meaning of exactly the same type as the meaning of a VP with only an intransitive verb. Both are world/time-dependent functions of type $\langle e, t \rangle$ (put differently, they characterize sets of individuals). Because of this, we also now have a simple way to give the semantics for conjoined VPs without having to assume that these really involve hidden S conjunction. For expository simplicity, we ignore the possibility of allowing the denotation of a VP to be a partial function from the domain of individuals, and so assume that each VP maps each individual into either true or false.⁹ Consider the phrase structure rule in (6) which introduces VPs conjoined with *and*. (Keep in mind that this will eventually be generalized to allow coordination of all categories.) Up until now we have in our fragment only a semantics for *and* that conjoins sentences (TR-1), but the discussion in 5.1 led to the conclusion that there is VP conjunction as well. Direct Compositionality requires a semantics to go with (6), and we are now in a position to see what this

⁹ As noted several times, a possible way to model presupposition failure is to use partial functions. To use an example we have seen before, we might ultimately want to treat the presupposition in a predicate *stopped smoking* by saying that it is undefined (at a world/time pair) for all individuals who never smoked. If this is correct, the rule for conjunction given below will need refinement.

semantics is. Thinking in set terms, the contribution of *and* is simply intersection. (Ultimately one would hope for single *and* for sentences, VPs, and other expressions; this is accomplished in Chapter 12.) In other words, consider a sentence like a slight variant of (3c) *Lee meditated and fed Porky*. The intuition is easiest to get in set (rather than function) terms. Since $[[\text{feed}]]$ is of type $\langle e, \langle e, t \rangle \rangle$, it combines with $[[\text{Porky}]]$ (some individual) to give the characteristic function of the set of individuals who fed Porky. $[[\text{meditated}]]$ also characterizes some set. These two sets can intersect; and the result is the set of individuals in both the pig-feeding set and the meditating set. This is exactly right; when this combines with Lee, the sentence says that Lee is in this set—i.e., the set which is the intersection of the meditators and Porky-feeders. By the definition of intersection, it follows that Lee is in each set. Notice, then, that with ordinary individual-denoting subjects (like *Lee*) the truth conditions for the case of conjoined VPs are the same as the truth conditions for the case of conjoined sentences, as in *Lee fed Porky and Lee meditated*. But this follows from the semantics that intersects the two VP meanings, and so the correct truth conditions are arrived at without having to pretend that there are really two sentences.

To formalize the rule for conjoined VPs, note first that since we are not considering the case of partial functions here we can safely go back and forth from talking about sets of individuals to the corresponding characteristic function of this set and vice versa. Given a function f of type $\langle x, t \rangle$ (for any set x) we have introduced the notation f_S to indicate that subset of x mapped to 1 by f . Now take any set z which is a subset of some set x . We now introduce the notation z_F to indicate the function of type $\langle x, t \rangle$ that characterizes z (i.e., the function mapping all members of z to 1 and all other members of x to 0). Using these notational devices, we can state the VP conjunction rule (extensionally):

TR-7. If α is an expression of the form $\langle [a], \text{VP}, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], \text{VP}, [[\beta]] \rangle$, then there is an expression γ of the form $\langle [a\text{-and-}\beta], \text{VP}, ([[a]]_S \cap [[\beta]]_S)_F \rangle$.

Thus the meaning of the conjoined VP is the function characterizing the intersection of the two sets characterized by the daughter VPs. The rule introducing *or* is similar, where here the relevant semantics is set union:

TR-8. If α is an expression of the form $\langle [a], \text{VP}, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], \text{VP}, [[\beta]] \rangle$, then there is an expression γ of the form $\langle [a\text{-or-}\beta], \text{VP}, ([[a]]_S \cup [[\beta]]_S)_F \rangle$.

5.5. Take a universe with three individuals $\{a, b, c\}$. Take a world w_1 such that $[[\text{kiss}]](w_1)$ is as shown in (17) above. Assume further the following: $[[\text{Juliet}]](w_1) = a$ and $[[\text{Romeo}]](w_1) = c$. Moreover $[[\text{sleepwalk}]](w_1)$ is the following function:

$$a \rightarrow 0, b \rightarrow 1, c \rightarrow 1.$$

Take the sentences in (i) and (ii):

(i) Juliet sleepwalked and kissed Romeo.

(ii) Romeo kissed Juliet or sleepwalked.

Show the full syntactic and semantic composition of this sentence extensionally with respect to w_1 . In other words, show the value of each expression at w_1 which the syntax proves well-formed. (Use the model in (12) in Chapter 4 for how to display this.) For all of the VPs, show these both as sets and as the functions which characterize those sets. What is crucial is to see why the sentence doesn't need to be converted into two separate sentences in order to arrive at the correct semantics for the whole; showing out the extension of each of the well-formed VPs in set terms helps to elucidate that point.

This account will eventually be refined in several ways. First, *and* and *or* will ultimately be listed in the lexicon (so they themselves have a meaning) and will be members of a single syntactic category. Second, there appears to be evidence that both *and* and *or* actually take their arguments one at a time such that the structure of, for example, *walks and chews gum* is actually $[[\text{walks}[\text{and}[\text{chews gum}]]]]$ (see, e.g., Munn 1993). Note that this means that $[[\text{and}]]$ is (thinking in set terms) the Curry'd version of the intersection operator. (The interested reader can use prose to state just exactly what this function is.) Both of these desiderata are accomplished in the revision in the next chapter. And finally, as noted above, conjunctions like *and* and *or* are cross-categorial: for any two expressions $[\alpha]$ and $[\beta]$ of the same category X, there is also an expression of the form $[\alpha\text{-and-}\beta]$. The complex expression itself is of category X; we know this because it has the same distribution as any other expression of category X. Ultimately then a single *and* and a single *or* should be listed in the lexicon which serves to conjoin expressions of any category. A full generalization is given in Chapter 12 (and section 13.4.4.2

contains yet another way to accomplish the generalization). But for the moment we will not collapse S conjunction (TR-1) with VP conjunction (TR-7). Note, though, that while the semantics for VP conjunction and for S conjunction look at first blush to be completely different, they have something in common. Recall that the value of an S is a function characterizing a set of worlds; the fully intensional version of the *and*-rule in the case of S conjunction also intersects these sets.

5.6. In light of the remarks above, it is tempting to try to collapse the two conjunction rules as follows:

FR-3 If there is an expression α of the form $\langle [a], X, [[a]] \rangle$ and an expression β of the form $\langle [\beta], X, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [a\text{-and-}\beta], X, ([[\alpha]]_S \cap [[\beta]]_S)_F \rangle$, for X ranging over S and NP .

Aside from the fact that ultimately we want to allow in conjunction of other categories besides just VP and S, there is a “cheat” embodied in FR-3. Why is it cheating to try to collapse the case of VP conjunction and S conjunction in the way done in FR-3?

Straw Man’s final problem was that his conjecture that VP conjunction was secretly S conjunction gave (without additional complexities) the wrong meanings for sentences like (14a) and (15a) where the subjects are quantified:

(14) a. Some pig chased the wolf and grunted.

(15) a. Every pig chased the wolf or grunted.

Under the two-sentence analysis of these, these should have the same meaning as the corresponding sentences where the subject is repeated, and yet they do not. But we cannot truly boast that the Curry’ed transitive verb solution solves all of Straw Man’s problems until it is shown that this problem is avoided here. In fact, we are not yet in a position to do that: but this is only because the meaning of NPs with quantifiers has not yet been discussed. We postpone this until Chapter 10. It turns out that armed with a meaning for expressions like *every man*, etc., the meaning (or at least preferred meaning) of (14a) and of (15a) is exactly as predicted. (It will also be shown that the secondary meanings can be derived.) At this point we can only say “stay tuned.”

*5.5. A note on three-place verbs

English has many verbs that occur with two constituents within the verb phrase, as in *gave the new toy to Mitka*, which consist of the sequence V-NP-PP (we will call the verb V_3 here). There are also cases of verbs which take sequences of two NPs such as another version of *give*: *gave Mitka the new toy*. (The relationship between the two versions of *give* is much discussed in the generative grammar literature; here we focus only on the version that takes an NP and PP, but see also Exercise 9.10.) And some verbs take NPs and sentential-like expressions (which we will call CP) as in *told Chicken Little that the sky is falling* and so forth.

Here we focus on the V-NP-PP verbs such as *give*. Before going further we need to say something about the meaning of *to Mitka*. Assume that *to* functions much like a case marker and contributes nothing to the meaning, so that *to Mitka* means the same as *Mitka*. Returning to the full VP *gave the new toy to Mitka*, there are four logical possibilities for how this is put together in the syntax (indeed, all four possibilities have been suggested):

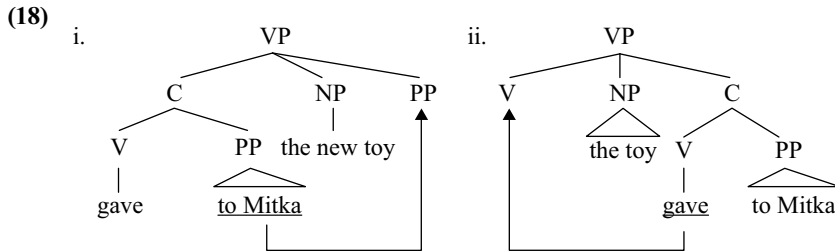
1. The VP is “flat”: in phrase structure rule terms there is a rule of the form:
 $VP \rightarrow V_3 \text{ NP PP}$ (see, e.g., Gazdar, Klein, Pullum, and Sag 1985).
2. The NP and PP actually form some sort of constituent; call it Z and there are two rules:
 $VP \rightarrow V_3 Z$ and $Z \rightarrow \text{NP PP}$ (see Kayne 1984).
3. *give* and *the new toy* form a constituent which then combines with a PP (as does the simple verb *dash*). The new rules would thus be something like:
 $VP \rightarrow V_4 \text{ PP}$ and $V_4 \rightarrow V_3 \text{ NP}$.

This means that *give* and *the new toy* combine to give a complex verb of the same category as things like *dash*; *gave the new toy* then combines with the PP *to Mitka* and the result is a VP (see, e.g., Dowty 1988).

4. The last possibility is one which requires additional apparatus in the syntax, as the syntax involves more than just concatenating expressions. This would be to assume that *give* actually first combines with *to Mitka* and the resulting expression then combines with *the new toy*. Versions of this (either for this case or for related cases) are proposed in Chomsky (1957) (for a VP like *consider George a fool*), and explored in much

subsequent Categorical Grammar literature (Bach 1979, 1980, Dowty 1982, and others), within GPSG by Jacobson (1987), and then again in Government and Binding theory in Larson (1988).

The basic idea of the proposals in 4 above is that *gave to Mitka* in the expression *give the toy to Mitka* is a *discontinuous constituent*. There are two broad classes of ways to implement this intuition. One makes use of two levels of representation in the grammar, and posits a movement rule to break up the constituent. This in turn has two possible implementations. (i) The basic structure is as in (18i) and *to Mitka* moves to the right as shown there, or (ii) the basic structure is as in (18ii) and *give* preposes (we suppress details of the actual proposals, and we arbitrarily adopt the label C for the expression that is the discontinuous constituent).



Chomsky (1957) proposed a version of (i) (although for a different set of cases), and Larson (1988) proposes a version of (ii).

A rather different way to implement the basic intuition is put forth in Bach (1979, 1980). This relies on the observation that the existence of discontinuous constituents does not inevitably lead to the conclusion that there is more than one level of representation. Rather, the syntactic rules combining two expressions might perform operations other than just putting one (or more) expressions next to each other. Perhaps there are also cases where one expression is infixed into another.¹⁰ (Note that

¹⁰ A common misconception is that the existence of “discontinuous constituents” requires more than one level of representation because a sentence must be represented as a tree, and trees do not admit of discontinuous constituents. If the grammar contained an infixation operation, the representation of the sentence would involve an object with crossing branches—not a tree. But the underlying assumption—that trees are sacred objects—makes little sense; a tree may or may not

morphological processes include infixation.) Bach dubbed the process in syntax as *Wrap*, and proposed that *give* first combines with *to Mitka* to form a complex transitive verb *give to Mitka*. Further, any complex transitive verb combining with its object (here, *the new toy*) does so by “wrapping” around its object. Bach’s particular formulation of the operation referred to word boundaries: Wrap places one expression after the first *word* of the other. Wrap operations have been formalized in other ways as well. For example, Pollard (1984) suggests that the grammar refers not just to unanalyzed strings but *headed* strings. Each string has a distinguished head, and Wrap places the infix after the head of the other expression (or, in some languages, before the head of the other expression). If there are Wrap operations, the remarks made earlier regarding the role of structure in grammar are not quite correct: the grammar must keep track of just enough structure within an expression as to specify the infixation point. But none of the Wrap proposals requires the grammar to refer to structures as rich as full trees. We will formalize a variant of Pollard’s proposal in Chapter 6.

This fourth solution is the most complex in terms of the syntax: it requires a Wrap operation (or a movement rule). Nonetheless, there seems to be some evidence for it (see section 15.4.2). To the extent that we deal in this book with three-place verbs, the Wrap solution is what we will (tentatively) adopt. (Note that this is compatible with the general architecture assumed here while the movement solution is not.) And even though the evidence for Wrap in the case of English is not entirely conclusive, there is independent evidence that grammars containing only the equivalent of context-free phrase structure rules will not be enough for other languages (see n. 5 and, e.g., Shieber 1985). Wrap operations have

be the appropriate representation as that depends entirely on what we discover is the correct rule system. (A tree is a sensible representation of how the rules work in a context-free phrase structure grammar; it is not a sensible representation in other systems.) Notice that if there are multiple levels of representation then it is also the case that a tree is not the appropriate representation for a sentence. Rather, a *sequence* of trees is. There is no particular reason to prefer a sequence of trees to some other kind of object. The important point is that *representations* represent the rule system(s); the task is to discover the kinds of rules employed by natural language.

been shown to be useful in the analysis of some of the key cases that lead to this conclusion.¹¹

In any case, the main interest here is in how the semantics works. Given the hypothesis of Direct Compositionality, each of the four logically possible analyses of three-place verbs sketched earlier leads to a slightly different way to flesh out the semantics associated with the syntax. Here we consider only the possibilities labeled (3) and (4) above. While we have seen that (4) is syntactically more complex, (3) and (4) are equally straightforward in terms of the semantics. Either way, the idea is that—once again—the verb in the syntax is combining with its arguments one at a time. And since both of these are individuals, the resulting verb is of type $\langle e, \langle e, \langle e, t \rangle \rangle \rangle$. The meanings would be different depending on whether we adopted (3) or (4). If (3) is correct, *[[give]]* as a function maps an individual which serves as the gift to a function taking the recipient to a function from givers to 1 or 0. If (4) is correct, the recipient is the first argument, the gift is the second argument, and the giver is the last argument in.

- *5.7.** (i) If (1) were correct, what might be its associated semantics?
 (ii) One way to give a semantics that goes along with (2) would be similar to the way you (hopefully) constructed for (i). Make this explicit.

¹¹ One such example is the analysis of “cross-serial dependencies” in Dutch given in Pollard (1984). The key phenomenon was discussed in Huybregts (1976). To illustrate the basic phenomenon, we use subordinate clauses (because Dutch, like most other Germanic languages, always has the main verb in second position in main clauses, which complicates the situation). Thus we find embedded sentences in Dutch like the following:

- (i) ... dat Jan Piet Marie Cecilia zag helpen laten zwemmen
 that Jan Piet Marie Cecilia saw help make Swim
 “that Jan saw Piet help Marie make Cecilia swim”

This consists of a series of NPs and a series of verbs, and the “subject” of the *n*th verb in the verb chain is the *n*th NP in the NP chain. (This is not an isolated example but a systematic phenomenon.) While a set of context-free phrase structure rules could literally give the relevant string set it cannot give it in a way which—assuming a Direct Compositional view—will give a sensible semantics. Pollard (1984) showed that supplementing concatenation rules with Wrap can account for this.

6

Categorial Grammar

6.1. Basics	89	6.5. Further generalizing the rules	98
6.2. Syntax/semantics correspondence	92	6.6. <i>and, or</i> , negation, and case-marking prepositions	100
6.3. Refinements	94	6.7. Summary: The fragment so far	104
*6.4. Some further observations about syntax: Case marking and agreement	96		

While the goal of this book is not to develop a detailed theory of syntax, a work on the syntax/semantics interface obviously needs to be rooted in some theory of syntax, even if only for expository purposes. We thus now develop (one version of) the theory of Categorial Grammar, albeit in a somewhat rudimentary fashion.¹ The reason for adopting Categorial Grammar is that it provides an elegant account of how the syntax and semantics work together. However, one should bear in mind that most of the remarks in the rest of this book on the syntax/semantics interface could be imported

¹ Categorial Grammar actually predates other versions of generative grammar, and was first developed in Ajdukiewicz (1935). Montague (1973) adopted a version of Categorial Grammar as his syntax for much the same reason as we are using it here: it provides a transparent fit between the syntax and the semantics. There are a number of different versions of Categorial Grammar. See, for example, Steedman (1987, 1996) which has often gone under the name Combinatory Categorial Grammar (the basic version adopted in this text is quite close to although not identical with this). Much recent work in the general framework has gone under the rubric of “Type Logical Grammar”; see Carpenter (1998) for an introduction.

into other theories as well, with of course some modification. Indeed some of the proposals in this book are based on ones originally developed within the theory of Generalized Phrase Structure Grammar (Gazdar, Klein, Pullum, and Sag 1985) (which adopted the semantics of Montague 1973). Head-Driven Phrase Structure Grammar (HPSG) (see, e.g., Pollard and Sag 1994) incorporates a rather different theory of semantics, but nonetheless maintains essentially the same architecture as the one here for the syntax/semantics interface. And while Government–Binding Theory and Minimalism both have very different views of the organization of the grammar, many of the basic insights to be discussed later regarding the semantics carry over to or even were developed within these theories. As we do not wish the main body of results reported in this book to be locked into a single theory of syntax, the reader familiar with other theories is strongly encouraged to translate the results into the terminology of those theories from time to time where possible.

6.1. Basics

A fundamental premise of Categorical Grammar (hereafter, CG) is that a syntactic category label is an expression of its distribution. The grammar contains a small set of primitive categories and defines other categories from these. Let us take the primitives to be S, NP, N, PP, and CP.² The fact that some of these (NP, PP) are of the form “XP” (standardly used to mean *X-phrase*) while others (S, N) are not is arbitrary and is just a bow to tradition: there is no distinction between categories which typically contain only one lexical item and those which typically contain more. Indeed this was noted already in section 4.1. There, complex expressions like *the disobedient husky* were treated as being of the same category (NP) as simple names like *Mitka*.

In addition to the primitive categories, we recursively define additional categories as follows:

- (1) If A is a category and B is a category, then A/B is a category (this category is pronounced as “A slash B”).

² CP is the name for the category of expressions like *that Mitka howled* in a fuller sentence like *Sabrina thinks that Mitka howled*. Thus this is distinct here from S (it is an S combined with *that*), although we will assume that *that Mitka howled* has a meaning of type <s,t>, just as does S.

The intuition behind labeling some expression with the category A/B (for some A and some B) is that such an expression combines with an expression of category B to give an expression of category A. In other words, the name of a category encodes what it syntactically combines with (as a sister) and the category of the resulting (mother) expression.

Obviously missing from the above list of primitives is the category “VP.” This is because any expression of this category can be recast as something which combines with an NP to give an S;³ the traditional “VP” is thus renamed as S/NP. We will, however, often use the label VP as a convenient abbreviation for S/NP. Consider then the case of transitive verbs such as *kiss*, *love*, *chase*, etc. Each of these combines with an NP to give a VP, and so each is of category (S/NP)/NP. A verb like *dash*, on the other hand, which takes a PP complement within the VP (*The frightened fox dashed into a hole*) is of category (S/NP)/PP. In other words, the complement that a verb takes within the VP is encoded into its category. What about intransitive verbs such as *sing*, *walk*, etc.? We now officially adopt the view that the grammar does not distinguish between complex phrasal expressions and expressions that have the same distribution but happen to consist of just a single lexical item.⁴ Since *walk* has exactly the same distribution as the complex expression *chew gum* and the same type of meaning (as was demonstrated at great length in the preceding chapter) a simple intransitive verb like *walk* is also of category S/NP. The astute reader will notice that we have suddenly glossed

³ In fact a variety of other constituents besides NPs can serve as subjects in English. This includes “CPs” (sentences introduced by *that*) as in *That Sarah was the nominee was extremely surprising*, and infinitive VPs as in *To ski while being pulled by a dog is tremendous fun*. Arguably verbs do not impose any idiosyncratic restrictions on the category of their subject in English: unlike in object position the category with which they occur seems to be predictable purely semantically. (See Grimshaw 1982 for discussion.) It may well be then that a “VP” should really be defined as S/X for X a variable over a set of categories, but we will not pursue this here.

⁴ There is one situation in which English grammar seems to be sensitive to whether an expression has more than one word in it. Simple adjectives (like *red*) occur prenominal, but complex adjectives (*happy that Bill left*) occur only post-nominal when they modify nouns. Of course this means that the two do not have exactly the same distribution so they could be given a different category. But in all other respects they are the same, so we will not take that route. We leave it open as to how exactly the grammar refers to this difference; see section 8.4 for a treatment of adjectival nominal modifiers.

over an observation made earlier: the traditional notion of “verb” is motivated by morphological considerations. As things are set up so far, *kiss* and *walk* have nothing in common: one is of category S/NP and the other of category (S/NP)/NP. But these items are treated alike by the morphology: they all take past-tense morphology, they can take the suffix *-ing*, and so forth. This will be addressed shortly.

Thus the intuition is that (like in many other theories), the combinatory rules can be stated in a very general form. Most of the information about what kinds of complements an item can combine with has to be listed in any case as part of the lexical entry. Given that, the rest should follow from some very general rule schemata. This is much like the idea in, for example, classic X-bar theory but the particular rule schemata will be quite different from those adopted in X-bar theory. (Readers unfamiliar with X-bar theory can consult an introductory syntax text for discussion, but it will play no role in this text.)

As noted above, the basic idea is that if something is of category A/B, it can combine with an expression of category B to give an A. This however is not quite enough: the category so far does not specify whether the B expression goes to the left or to the right of the A/B expression (or, in light of the remarks in section 5.5, possibly as an infix). We thus elaborate the categories in such a way that they encode that as well. Using a notation not entirely standard in Categorical Grammar, an expression of category $A/_RB$ is something which combines with a B to its right to give an expression of category A, and $A/_LB$ takes the B to its left.⁵ (Cases involving Wrap—if they exist—are dealt with in the fragment summary at the end of this chapter.) The recursive definition of categories is then expanded as follows:

⁵ There are two other common notational systems for this which unfortunately collide. In one system (cf., Lambek 1958 and much work in Type Logical Grammar) A/B means something which takes a B to its right to give an A, while $B\backslash A$ is something which takes a B to its left to give an A. In Steedman (1987, 1996) and related work A/B continues to mean something that takes a B to its right to give an A, while $A\backslash B$ is something which takes a B to its left to give A. Thus the two systems collide on the meaning of $X\backslash Y$: in one the result is X and in the other the result is Y. It is because of this potential confusion that we adopt the subscripts. Subscripts are also convenient if the grammar turns out to include a Wrap operation; a “Wrapper” can be notated as $A/_WB$. (This is what is known in morphology as a circumfix, as in the German morpheme *ge...en* indicating past participle. Something which is an infix would be an $A/_IB$.)

- (2) If A is a category and B is a category, then $A/_RB$ is a category and $A/_LB$ is a category.

The intuition behind these categories has already been discussed, but technically all we have in our grammar so far is a list of lexical items and categories. Thus the grammar also includes two general rule schemas, which replace rules TR-5' and TR-6 in the earlier fragment (for now, the semantics is not spelled out: “...” is used as a placeholder). We will label these rules as R-1a and R-1b. Lest it appear that we are cheating by counting this as one rule when there really are two, later discussion (section 6.5) will collapse them into a single rule.

R-1a (partial): If α is an expression of the form $\langle [a], A/_RB, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [a-\beta], A, \dots \rangle$.

R-1b (partial): If α is an expression of the form $\langle [a], A/_LB, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [\beta-a], A, \dots \rangle$.

At this point these two rules replace only two rules in the fragment constructed earlier, and so there is no sense in which these rules are more general. But that is obviously only because the fragment so far is rather small: there will be many other applications of the rules R-1a and R-1b.

Speaking informally for the moment, there is a sense in which an expression with a category of the form $A/_LB$ is a function from strings to strings. (This will be formalized in 6.5.) Thus *sings* maps *Madonna* to *Madonna sings*. Similar remarks hold for right-slash categories. Hence, when an A/B combines with a B we refer to the former as the function and the latter as the argument. Unfortunately the syntactic notation reverses the domain and co-domain from the semantic notation; A/B takes an expression of category B and returns one of category A , while $\langle a, b \rangle$ is a function taking something from domain a and returning a value in co-domain b . This confusion is an unfortunate accident of history which simply takes some getting used to.

6.2. Syntax/semantics correspondence

What about filling in the semantics? For the two instances of the schemas that we know of (combining VPs with subjects and transitive verbs with objects) the semantics involves applying a function to an argument. In order

to state this generally, the correspondence between syntactic categories and semantic types needs to be made explicit. Each primitive syntactic category corresponds to some semantic type, i.e., all expressions of some primitive syntactic category have as their meanings some member of the same set. The extensional type of *S* is *t*—in other words, the value of any *S* is a member of $\{1,0\}$. Its intensional type is $\langle s,t \rangle$. The extensional type of *NP* is *e* (its intensional type $\langle s,e \rangle$). We also see that *S/NP* is of type $\langle e,t \rangle$ (intension: $\langle s,\langle e,t \rangle \rangle$) and *(S/NP)/NP* of type $\langle e,\langle e,t \rangle \rangle$ (intensional type: $\langle s,\langle e,\langle e,t \rangle \rangle \rangle$). Assume then the following correspondence between syntactic category and semantic type: for an expression of category *A/B*, its extension is a function from the extensional type of *B* to the extensional type of *A*. (There is a caveat: some expressions are functions from intensions not extensions; this is the subject of Chapter 19.) This correspondence holds regardless of the directional feature on the slash. Note incidentally that while this assumes that all expressions of a given syntactic category have meanings of the same types, the reverse does not hold. Thus *A/LB* and *A/RB* are syntactically different categories, but have the same type of meaning. Put differently, word order has no reflex in the semantics and is purely a syntactic phenomenon. And Chapter 8 discusses two other syntactic categories besides *S/NP* whose meaning is of type $\langle e,t \rangle$. The set of syntactic categories, then, is richer than the set of semantic types.

Having made explicit the correspondence between syntactic categories and semantic types, it is now obvious how to fill in the “...” part in the above rules. We give the full rules extensionally, and we will refer to R-1a as **r-app** and R-2a as **l-app**:

- R-1a: If α is an expression of the form $\langle [a], A/RB, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [a-\beta], A, [[a]]([[\beta]]) \rangle$.
- R-1b: If α is an expression of the form $\langle [a], A/LB, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [\beta-a], A, [[a]]([[\beta]]) \rangle$.

6.1. Give INT-R-1a, using the model for stating this shown in section 3.3.

6.3. Refinements

There are some aspects of this system that might strike a reader familiar with syntactic theory as objectionable, but these objections are easily addressed. First, we don't want to conclude that every intransitive verb is listed in the lexicon of English as having category $S/_LNP$, where the directional feature (L) is stipulated on a case-by-case basis. For it is a perfectly general fact about English syntax that subjects (which can be defined here as the last argument in forming an S) go to the left. Similarly for the right slash: all transitive verbs take their objects to the right and this should not be listed lexical item by lexical item. And the generalization is even broader: verbs in English that occur with a single complement within the VP take it to the right regardless of the category of that complement. In fact, most instances of expressions of category A/B —provided that A is not just the simple category S—combine with things to the right. Prepositions, for example, are of category PP/NP and they take the NP to the right; *the* is listed as NP/N and it takes the noun to the right, etc. In many syntactic theories, all of these facts are subsumed under the observation that English is a *head-first* language. (Verbs are thus seen as the head of the VP; determiners as head of the NP (often called DP for just that reason), prepositions as the head of the PP, etc.)

While the version of CG developed here has no formal notion of head,⁶ the basic generalizations that go under the rubric of “head first” are easily accounted for. Note that there is a rough correspondence between the standard notion of head and the notion here of a function. Thus an expression α which is of category A/B can (roughly) be thought of as the head of an expression formed by combining it with a β of category B. To the extent that in general heads precede their complements, the generalization in CG can be recast as the observation that slashes in English are generally marked with a R-feature. An exception, of course, is the slash in S/NP (or S/X for other categories which can appear as subject).⁷

⁶ Such a notion could be added in if need be, though it will play no role in this text except possibly for the formulation of Wrap operations (see section 5.5).

⁷ In standard X-Bar theory, the fact that subjects precede VPs is accounted for by not treating the VP as a “head” but rather treating the subject with a separate label: “Specifier.” There is no corresponding notion of a Specifier in (at least this version of) CG.

Assume, then, that a lexical item is listed in the lexicon with an underspecified category, and that there are general rules adding further information to the category. We will use the term *lexeme* to refer to that which is listed in the lexicon. Thus the directional feature is not listed on a case-by-case basis on individual lexemes: *kiss* is just listed in the lexicon (S/NP)/NP. But only a more fully specified version (with a slash directional feature) can actually combine with another expression. Hence the two rules in (3) apply in English, where X and Y are variables over any category. These are displayed informally but the intent should be clear (the notation ... (S/X) ... means any occurrence of this material; this might be contained within a more complex category):

- (3) a. ... (S/X) ... b. ... (X/Y) ...
 ↑ ↑
 L R

If (b) applies after (a) the effect will be that a right slash is the default. (Thus a rule like this applies only to slashes not already marked with a directional feature.)

We add one additional directional rule. While modifiers have fairly free placement in English, a rough rule of thumb is that they occur to the right of what they modify. *Modifier* can be given a technical definition in CG: anything of category X/X is a modifier. Hence to the rules above can be added (c), which also applies before the (default) right-slash rule above (the order of application of (a) and (c) is not relevant):

- c. ... (X/X) ...
 ↑
 L

A second point to address is the fact that—despite the earlier observations about the morphological commonality of “verbs”—we seem to have no such notion at our disposal. Similar remarks would ultimately hold for nouns and adjectives. While ordinary intransitive nouns are of category N, nouns that take PP complements (including *relational nouns* like *aunt* which will be discussed in Chapter 8) are presumably of category N/PP. While beyond the scope of the semantic treatment here, there are also nouns that take CP complements (*belief that the world is round*) and hence presumably of category N/CP. Yet nouns like this all have a morphological commonality (they all can have plurals). Indeed—as noted in Chapter 4—the

traditional parts of speech are motivated by morphological considerations. But there is no problem here: it is perfectly easy to recover the traditional notions in terms of the ultimate *result* category. A *verb* is something whose ultimate result category is S (this will be slightly refined in section 8.1). A *noun* is something whose ultimate result category is N. Morphological processes are presumably sensitive to classes defined in this way.


***6.4. Some further observations about syntax: Case marking and agreement**


Once one starts thinking of a syntactic category A/B as the information that “my sister will be a B and my mother an A,” there are other interesting consequences for syntax. As in most other theories of syntax, we can refine the basic categories given above to actually be feature bundles (let S, for example, be one feature in the bundle). Gender, number, case, and so forth are additional features on these labels. In CG, phenomena like case marking, agreement, and so forth are naturally handled as *selection* for elaborated categories (rather than as a process transferring features onto actual linguistic material). For example, *walks* is presumably S/NP[SING] while *walk* is S/NP[PLUR].⁸ Presumably there is a single entry in the lexicon which is not specified for singular or plural, and morphological rules apply to the lexeme WALK of category S/NP and map this to [walk] of category S/NP[PLUR] and [walks] of category S/NP[SING]. There is also a semantic effect of this process. It was briefly noted in Chapter 4 that plurals can be thought of as individuals of a special type, and so the meaning of WALK is a function defined for the full set of individuals (both singular and plural). The mapping of this to the more fully specified version S/NP[SING] would remove the plural individuals from the domain of this function (and the reverse for the plural version).

Similarly, case marking is really case selection. Take a language with robust case marking but where the case marking is quite predictable—

⁸ One might well want a more sophisticated theory of features where a feature is a name and a value, and so there could be a single feature named NUM whose possible values in English are SING/PLUR. In languages with a singular/dual/plural distinction, the NUM feature has three possible values. See Gazdar, Klein, Pullum, and Sag (1985) for a well-worked-out theory of features along these lines.

suppose for example all subjects are nominative and all objects accusative. Then all the transitive verbs in the language are actually of category (S/NP [NOM])/NP[ACC]. As for the case of English word order, one would not want to conclude that this information is listed in the lexicon on a case-by-case basis. Hence, let the verbs be listed in underspecified form, with the following two rules to add the case features to the NP slots:

- (4) a. ... (S/NP) ... b. ... (S/NP) / NP
- 
 NOM


 ACC

Even more interesting is the situation in a language like German in which the bulk of the verbs assign a predictable case to their objects (hence this is called *syntactic case*) while a small number assign a different case. For example, the verb *helfen* “help” assigns dative case, and it is generally assumed that this is a lexically idiosyncratic fact which must be learned by German speakers on a case-by-case basis (this is thus often called *lexical case*). On the other hand, speakers do not need to learn a list of verbs assigning accusative case, for that is the default. Here we assume that German contains the two rules in (4), but that *helfen* is listed in the lexicon as (S/NP)/NP[DAT]. As was the situation with the directional features, case features are assigned only to slots not already specified for a case feature. Hence they can add but never change the case feature on a slot. Thus phenomena like predictable (syntactic) case vs idiosyncratic (lexical) case are easy to handle.

***6.2.** The remarks above show how one might ensure that NPs with the appropriate case markings appear in the correct position, but there remains the question of ensuring that the correct internal structure of an NP with a certain case marking. In some languages the actual case marking shows up in the morphology of the noun, in some it shows up in the determiner, and in some both.

Consider the case of an imaginary language in which the noun shows the case marking. We will call this language English', and imagine then that an NP in subject position which translates as English *the dog* is, in English', *the dog-us*. In object position the NP would be *the dog-um*. Assume (following, e.g., Gazdar, Klein, Pullum, and Sag 1985) a system as noted above where Case features can be analyzed as a feature name

(cont.)

and a feature value: the name is CASE and the possible values are NOM and ACC. Then we take *dog-us* to be $N[\text{CASE:NOM}]$ and *dog-um* to be $N[\text{CASE:ACC}]$.

Give the category for *the* in English'. You may need to make up some notational conventions, or you may be able to borrow one from phonology. The interested reader might also formulate the category of the suffixes *-um* and *-us*.

***6.3.** Consider the case of a language (like German) where the case marking shows up in the particular form of the Determiner. Thus (mixing up German and English) let our imaginary language be English'' which has NPs of the form *dem dog* in subject position and *den dog* in object position. Give the categories for *dem* and *den*.

6.5. Further generalizing the rules

At the moment, R-1a and R-1b are—despite the numbering—two separate rules. But their semantics is exactly the same, suggesting that there should be a way to capture that generalization. Of course, there are only two rules at issue, and so it is not quite so obvious on the basis of just these two that any major generalization is being missed, but if there are similar additional rules such the Wrap process briefly discussed in section 5.5 there would be yet another case of an A/B combining with a B with the same semantics.

There are two conceivable resolutions (which are actually not mutually exclusive). The first is what is known as *type-driven interpretation*, which was put forth in, among others, Klein and Sag (1985), who were also working within a Direct Compositional framework. While their theory of syntax was different than that taken here, their basic solution carries over directly. Thus they proposed that each individual rule (schema) is not listed in the grammar with the semantic part, but that the semantic part can be filled in with the following principle (their terminology is slightly different):

- (5) For any rule (schema) which allows an expression α whose meaning is a function in $\langle a, b \rangle$ to occur with an expression β whose meaning is a member of set a , the semantics is $[[\alpha]]([[\beta]])$.

The intuition, then, is that there are general principles in the grammar putting things together in the way suggested by the types, and so the

semantics could be pulled out of what is listed as part of the particular rules. (The type-driven idea has also been borrowed—with modification—into a large body of work in a non-Direct Compositional framework in which the trees are the input to the semantics. The idea under this view is that semantic composition of each node in the tree is predictable from the type of its daughters. See Heim and Kratzer 1998 for one implementation.)

But it is not clear that type-driven principles (of the sort envisaged by Klein and Sag) are ever needed given the generality of rules in Categorical Grammar. The other option here is to collapse R-1a and R-1b (and if there is a Wrap rule it would collapse with these as well). This is the strategy that will be adopted here, which is why the two rules were numbered as 1a and 1b.

In order to collapse these, we first formalize the notion introduced informally earlier: that syntactic categories correspond to functions from strings to strings. While the necessary formalization might at this point seem like a lot of trouble to go to just to collapse these two rules, it actually has significant other benefits which will emerge in later chapters. Hence, each category of the form A/XB (where X is either L or R) corresponds to a function. The category itself is the name of a set of expressions, but it corresponds to some function which takes two strings (one at a time) and returns a third. We will notate the corresponding function as $F_{A/B}$, and define them as follows:

$$(6) \quad \text{a. } F_{A/RB}([a])([\beta]) = [a\beta] \quad \text{b. } F_{A/LB}([a])([\beta]) = [\beta a]$$

We can now state a single R-1 (call it **app**):

R-1. If α is an expression of the form $\langle [a], A/B, [[\alpha]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle F_{\text{CAT}(\alpha)}([a])([\beta]); A; [[\alpha]]([\beta]] \rangle$ (where $\text{CAT}(\alpha)$ means the category of α).

For example, take the expression *sings* which is of category S/LNP . The function which corresponds to this category is one which applies to the string *sings* and then yields a function which applies to *Madonna* to give *Madonna sings*. So the fact that *Madonna* goes to the left of *sings* when they combine is a result of the fact that R-1 says that the string resulting from the combination of *Madonna* with *sings* is the value of the function $F_{A/LB}$ when applied to the string *sings* and then to *Madonna*. For expository convenience, we will often continue to refer to R-1a and R-1b separately.

6.6. *and*, *or*, negation, and case-marking prepositions

Before summarizing the new fragment, we are ready to make some modifications regarding *and* and *or*. In the fragment developed earlier, these words were not listed in the lexicon but were introduced syncategorematically. Moreover, both *and* and *or* were introduced by rules which were limited to just VP and S conjunction, but it is well known that any two expressions of the same category can combine (with either *and* or *or*) to give an expression of that category. That is, the resulting expression has exactly the same syntactic distribution as does each of the daughters.⁹ Each of the italicized expressions below illustrates this point:

- (7) a. The candidates spoke *to the moderator and to the audience*.
- b. Sam *cooked and ate* lentils for supper.
- c. *A friend and colleague* of mine was just named Dean of the Faculty.

So conjunction (and disjunction) should be generalized to be cross-categorical.

Moreover, in order to list *and* and *or* in the lexicon and to do so without introducing any new rules, we can assume that they combine with their arguments one at a time. In other words, both are of category (X/X)/X, for

⁹ One might ask whether coordination involves *only* the combining of two expressions of the same category or whether there are other cases of coordination. There is a large literature on this question, and many have concluded that there are other kinds of coordination. For one thing, there appear to be cases where categories apparently of different types conjoin, as in (i). Second, there are cases of so-called Non-Constituent Coordination (including what is sometimes known as Right-Node Raising) as in (ii):

- (i) Linda is a bank teller and proud of it.
- (ii) a. Mary loves and John hates model-theoretic semantics.
- b. Captain Jack served bananafish yesterday and lobster today.

But it will be shown in section 13.4.2 that within Categorical Grammar cases like (ii) of so-called “non-constituent conjunction” reduce to ordinary constituent conjunction (Dowty 1988). The cases in (i) are also not threatening to the generalization that all and only like constituents may conjoin with a more sophisticated notion of categories. This will not be pursued in this book, but the interested reader can consult Bayer (1996) for a CG treatment of these. A quick preview: this incorporates the notion of “union” categories and “intersection” categories. If categories name sets of strings, this becomes quite natural. See also Morrill (1994).

X a variable over categories. The claim that expressions like *Mitka and Kolya* do combine with their arguments one at a time actually has some motivation; see, e.g., Munn (1993).

6.4. Given the word order rules spelled out in (3a–c), what would be the directional features on each of these slashes? Show exactly how the grammar (in the syntax) thus derives expressions like *fed the pig and meditated* (ignoring, of course, the presence of the past-tense morphology). It is worth noting that—given certain assumptions shared in both Government–Binding theory (and its descendants) and in some versions of CG—there actually is evidence that the slash directionality features given here are correct. See Munn (1993) for the point phrased in GB terms, and Jacobson (1999) for a similar observation within the framework here.

We are, however, not yet in the position to give a fully general meaning for *and* and for *or* (partly for the reason hopefully discovered in Exercise 5.6, and also because there are cases where *and* and *or* connect expressions that don't denote sets, such as transitive verbs). But we do know that whatever the general meaning of *and* is, it has the effect of intersection for the case where it conjoins VPs. In this case, then, its meaning is the Curry'ed version of the intersection operator. So assume that the general meaning is such that whenever two set-denoting expressions are conjoined, the result is (Curry'ed) intersection. For now, then, we just write a complicated meaning for the lexical entry which says this. This meaning is spelled out in the fragment at the end of this chapter, and all of this is cleaned up in Chapter 12.

Note too that so far all of the other lexical items have as their meanings functions from worlds and times to extensions. Items like *and* and *or* seem different: these do not appear to have values that are in any way world- (or time-)dependent. Given two VPs combined with *and*, whatever world we are in is such that the result (in set terms) is the intersection of the extension of those VPs. That the value of $[[\text{and}]]$ is not world-dependent can be demonstrated as follows. Consider the sentence *The senior senator from Nebraska coughed and sneezed*. Imagine two worlds w_1 and w_2 such that this sentence is true at w_1 and false at w_2 . Then either *the senior senator from Nebraska* has a different referent in those two worlds, the set of coughers is different,

or the set of sneezers is different. If Nebraska's politics remain unchanged and the coughers and sneezers do too, this must have the same value at both worlds. This would not follow if $[[\text{and}]]$ was world/(time)-dependent.

But this is no different than the situation that we saw for proper names which (at least arguably) have a constant extension regardless of the world. Thus one way to keep the semantic combinatorics uniform, is to assume $[[\text{and}]]$ and $[[\text{or}]]$ are indeed functions from worlds to (roughly) Curry'ed intersection, but are constant functions. In fact, this is true in general for what are traditionally thought of as closed-class items.¹⁰ As we will see, prepositions, *the*, *be*, etc. also do not vary in their extensions from world to world. We will thus treat all of these as constant functions from worlds. Since these do not vary, we will from now on simply write their meaning extensionally. (The correlation between closed- vs open-class items and constant vs non-constant functions is not perfect, since the values of names are also (arguably) constant functions, but these are surely open-class items.) To a reader who feels that there is something rather clumsy and artificial about giving prepositions, *and*, etc. intensional meanings whose extensions are always constant across worlds, a possible modification of this will be very briefly explored in section 19.2.

We can also now give a somewhat more adequate treatment of English negation. Recall that the only kind of negation in the earlier fragment is introduced by TR-3, which appends *it-is-not-the-case-that* in front of a sentence and reverses its truth value. In reality, English negation generally involves either the word *not* after the first auxiliary, or *n't* as a clitic on the first auxiliary. But since a treatment of the English auxiliary system is beyond the scope of this book, we cannot here provide a fully accurate account of English negation. Still, we can do quite a bit better by pretending

¹⁰ For readers not familiar with this terminology: *closed-class* items are those whose syntactic category contains a limited number of items and does not easily admit of new such items (e.g., prepositions, determiners, conjunctions), while *open-class* items are those in the traditional categories like noun, verb, adjective, to which new items can be added with no difficulty (new nouns, verbs, and adjectives are coined daily). The former category is also sometimes called "function words." We make no commitment here to this distinction having any real theoretical significance since the difference in terms of ability for new items to be added is really a matter of degree and not of kind. Still, it is interesting to note that this traditional distinction does correlate roughly with the semantic distinction of having constant vs non-constant values across worlds (and/or times).

that *didn't/doesn't* are unanalyzed units which combine with VPs.¹¹ This word, then, is of category $VP/R\ VP$ ¹² and its meaning—in set terms—is set complementation. Hence $[[\text{didn't dance}]]$ is the function characterizing the set of individuals in the complement of the set characterized by $[[\text{dance}]]$. In function terms, $[[\text{didn't}]]$ takes some function $[[VP]]$ of type $\langle e, t \rangle$, and maps to 1 everything that $[[VP]]$ maps to 0 and vice versa.¹³

One final housekeeping detail concerns the case of so-called case-marking prepositions, as in *give the toy to Mitka*. We are leaving three-place verbs somewhat in limbo in the official fragment. But regardless of how these are treated, it is clear that *to* functions much like a case marker. That is, one cannot list the verb *give* just as selecting for an NP and a PP within the VP; it specifically selects for a PP with *to* as the preposition. A verb like *talk* also selects for a PP with *to*, as in *talk to the principal*. There are a few other

¹¹ There are cases where negation occurs with other kinds of expressions, as in *Not everyone left*. But unlike the case of *and* and *or*, *not* is not fully cross-categorical. Thus (1a) is not good, although we do get things like *not Mitka* in (ib):

- (i) a. *Not Mitka ate the rawhide.
- b. Kolya but not Mitka ate the rawhide.

The full range of English negation is rather complex, and so we confine ourselves to VP negation and introduce it by the single word *didn't*.

¹² For convenience, we have built the feature R into the lexical specification here to make sure that this overrides the general rule in (3c) according to which items of category X/X (i.e., modifiers) take their arguments to the left. In fact all auxiliaries might appear at first glance to need this kind of lexical stipulation, since they are all of category VP/VP but all take their VP complements to the right. But actually, the fact that auxiliaries (including *didn't*) take their complements to the right is unsurprising, for strictly speaking they are actually not of the simple category VP/VP. Each auxiliary specifies a particular morphological form for the item it combines with. *be*, for example, is something roughly like VP/VP[PROG] and so forth. Thus the actual directional feature on *didn't* will be specified by the general word order rules in the predictable way.

¹³ As usual, we ignore the possibility that some VPs denote partial functions on the domain of individuals; the usual assumption is that if some individual is not in the domain of $[[VP]]$ then it is also not in the domain of $[[\text{didn't VP}]]$. In other words, if an individual is not assigned a truth value by $[[VP]]$ then $[[\text{didn't VP}]]$ also assigns no value to that individual. In fact, a standard test for presupposition is that it “survives” negation, meaning exactly what is said above: if $[[\text{Bill}]]$ is assigned no truth value by $[[\text{stop smoking}]]$ because Bill never smoked, then $[[\text{Bill}]]$ is also assigned no value by $[[\text{didn't stop smoking}]]$.

prepositions in English which appear in places where they are not interchangeable with any other preposition. This includes *of* in *mother of Romeo* (see section 8.5), *on* in cases like *rely on the weather*, and *by* in *(was) chased by the wild boar*.

Syntactically, then, we need to mark the PP as a particular kind of PP. Borrowing a convention from Gazdar, Klein, Pullum, and Sag (1985) we will add a feature on such PPs, and—since there is nothing interesting about the choice of name of this feature—will make it mnemonic by using the name of the preposition as the feature. (Thus, *give* selects an NP and also a PP[TO].) The preposition *to* has the category PP[TO]/NP. What about the meaning of *to*, *of*, etc.? (We exclude the actual directional and/or locative uses of *to*.) In section 5.5 it was speculated that *to Mitka* means the same as *Mitka*. This means that the semantic type of the relevant PPs is type *e*.¹⁴ One might then determine that *to* is literally meaningless. But if so, we would need to add a new combinatory rule to the fragment to account for the semantic part of combining *to* with an NP. Thus instead of adding a new rule, it is perfectly straightforward to give *to* a real meaning: let it denote the identity function on individuals. Its semantic type is thus $\langle e, e \rangle$ (exactly as one would expect, given its syntactic category), and it takes an individual as argument and returns the same individual. That is the extensional meaning of *to* (and the other case-marking preposition); in light of the remarks above its full meaning is a constant function from worlds to the identity function.

6.7. Summary: The fragment so far

Here and henceforth we list the open-class and the closed-class items separately. Because we are not dealing with lexical semantics, there is in general

¹⁴ The semantics of PPs is one place where one might question the wisdom of assuming that all items of the same syntactic category have the same type of meaning. Cases like the ones at hand where the preposition is a case marker appear to be of type *e*. But PPs also occur both as noun and as VP modifiers, and even can occur with a few verbs that explicitly subcategorize for “adverbs” as in examples like *word the letter carefully/word the letter in a careful manner* (Bresnan and Grimshaw 1978). On the other hand, it may simply be a mistake to think that all of what we call “PPs” are of the same syntactic category: their distribution does wildly differ.

little to say about the meaning of the open-class items. For these, then, we list only their syntactic category and not their meaning.

Lexicon:

Open-class items:

NPs include: *Mitka, Balto, Barack Obama, the-disobedient-husky, the-pig, ...*

S/NP: *walk, talk, ...*

(S/NP)/NP: *kiss, likes, feeds, ...*

(S/NP)/PP[TO]: *talk, speak, ...*

(S/NP)/CP: *believe, think, ...*

Closed-class items:

(X/X)/X: *and, or*

[[and]]

if X is of type $\langle a, t \rangle$ (for any type a) [[and]] = Curry'ed intersection

if X = S, [[and]] is the following function:

$$\left\{ \begin{array}{l} 1 \rightarrow \left\{ \begin{array}{l} 1 \rightarrow 1 \\ 0 \rightarrow 0 \end{array} \right\} \\ 0 \rightarrow \left\{ \begin{array}{l} 1 \rightarrow 0 \\ 0 \rightarrow 0 \end{array} \right\} \end{array} \right\}$$

[[or]]: is left as an exercise

VP_R/VP: *didn't*; [[didn't]] takes a function that characterizes a set S, and returns the function characterizing the complement of S

PP[TO]/NP: *to*; [[to]] = identity function on individuals

CP/S; *that*; [[that]] = identity function on propositions

Word order rules:

i. ... (S/X) ...

↑
L

ii. ... (A/A) ...

↑
L

iii. ... (A/B) ...

↑
R

where (i) and (ii) must apply before (iii)

Combinatory rule:

Definition of functions corresponding to categories of the form A/B:

$$\text{a. } F_{A/RB} ([\alpha])([\beta]) = [\alpha-\beta] \quad \text{b. } F_{A/LB} ([\alpha])([\beta]) = [\beta-\alpha]$$

R-1 (app):

If α is an expression of the form $\langle [\alpha], A/B, [[\alpha]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle A/B_F([\alpha])([\beta]), A, [[\alpha]]([[\beta]]) \rangle$.

For expository convenience, we often break this into the two separate rules below:

R-1a: If α is an expression of the form $\langle [a], A/_R B, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [\alpha-\beta], A, [[a]]([[\beta]]) \rangle$. (call this **r-app**)

R-1b: If α is an expression of the form $\langle [a], A/_L B, [[a]] \rangle$ and β is an expression of the form $\langle [\beta], B, [[\beta]] \rangle$ then there is an expression γ of the form $\langle [\beta-\alpha], A, [[a]]([[\beta]]) \rangle$. (call this **l-app**)

**Possible extension for three-place verbs:*

A. If the Wrap analysis is incorrect, simply add to the lexicon items of category $((S/NP)/PP[TO])/NP$: give, donate, ...

B. If Wrap is correct:

Lexicon:

$((S/NP)/NP)/PP[TO]$: give, donate, ...

Word order rules:

Add: $((S/NP)/NP)/X$

↑
w

Note: X here ranges over any category. One could also allow it to range over any category including nothing; then ordinary transitive verbs would also be marked with a wrap slash. Presumably, though, the effect of wrap in that case would just be vacuous.

Formalizing the Wrap operation:

- a. Assume that each expression of the language is a string with an infixation point. We write such a string as $[x|y]$.
- b. Assume that each lexical item in English is a string of the form $[x]$. (This is actually needed only for a small set of items, but we will give it in full generality here.)
- c. Adopt a convention by which the infixation point in a new expression is inherited from the function. This can be done by refining the above definitions of the functions corresponding to syntactic categories as follows:

Definition of functions corresponding to categories of the form A/B :

Let α be some expression whose string is $[x|y]$. Then:

- a. $F_{A/_R B}([a])([\beta]) = [x|y-\beta]$ b. $F_{A/_L B}([a])([\beta]) = [\beta-x|y]$
- d. Define a function appropriate for $A/_W B$ as follows. Let α be some expression whose string is $[x|y]$. Then: $F_{A/_W B}([a])([\beta]) = [x|\beta|y]$

9

Interlude: The semantics of variables and the lambda calculus

9.1. First-order predicate logic (modified)	134	9.3. Naming model-theoretic objects	151
9.2. The lambda calculus	144		

This chapter detours from the main project of building a semantics (and syntax) for a fragment of English. Instead we pause to build a syntax and semantics for two languages that look very different from English. The first is basically (with modifications) first-order predicate logic which includes a treatment of quantifiers. The second takes the first language and enriches it to what is called the lambda calculus.

Since neither of these looks very much like English, one might wonder why engage in such a project. This might seem especially puzzling because we will see in the next chapter that English quantification is rather different from the treatment of quantifiers in first-order logic, so why bother with the latter? There are two good reasons to digress and build a semantics for these artificial languages. The first, and most immediate, is that the ultimate language developed in this chapter (the lambda calculus with quantifiers) provides a very useful way to name complex model-theoretic objects. The reader has undoubtedly noticed that some of the semantics in the preceding chapters has been stated quite awkwardly. This is fine as long as it is clear:

the language used to name model-theoretic objects is of no consequence for the theory and is only a tool for the linguist (the grammar itself simply pairs expressions with model-theoretic objects). But once the objects get sufficiently complex (e.g., functions with functions as their domains as well as co-domains) it is useful to have a simple way to write out these meanings without excruciating and/or ambiguous prose, and the lambda calculus allows us to do just this. We will, however, take the unusual step of trying wherever possible to also write these out in prose (however excruciating) as a reminder that these name actual model-theoretic objects and that the formulas have no theoretical significance.

The second reason for a detour into the semantics of these artificial languages is that the languages introduced here contain variables, and the semantics for the interpretation of variables translates rather directly into a standard view of the semantics of natural language pronouns. Variables (or their analogue) are also central under one common view of the semantics of relative clauses. But a caveat is in order: Part III develops two competing analyses of pronouns and relative clauses. The view using variables is embedded in an autonomous syntax approach, while the Direct Compositional alternative developed in Part III actually makes no use of variables at all (except as a notational device). And since this book is committed to direct compositionality, one might wonder why spend time discussing variables in terms of the theory of natural language. The reason is that the use of variables in semantics is so widespread that understanding the semantics of variables is a matter of basic semantic literacy. But in any case, it is time to develop more formally a metalanguage that we can use to name model-theoretic objects, since trying to continue with just English as the metalanguage would become more and more difficult.

9.1. First-order predicate logic (modified)

9.1.1. Variables

To depart minimally from what has been done so far, we begin by building a language a bit more English-like than is usual in treatments of first-order logic. We adopt a Categorical Grammar syntax, but with only S and NP as primitives. The correspondence between syntactic category and semantic

type remains the same, although intensions play no role here. Hence expressions of category *S* are of type *t*, expressions of category *NP* of type *e*, expressions of category *S/NP* of type $\langle e, t \rangle$, etc. There is no use of directional features on the slash as the syntax will put all functions before the arguments (making the language verb–object–subject in terms of word order). The noun phrases are all simple with no internal structure. To distinguish the expressions from actual English we notate them as *feed'*, *Fido'*, etc.

Lexicon:

NP: *Fido'*, *Porky'*, *Dumbo'*, ... (value of each is *f*, *p*, *d*, respectively)

S/NP: *walk'*, *chew-gum'*, *squeak'*, ... (each of which denotes some function in type $\langle e, t \rangle$)

(S/NP)/NP: *kiss'*, *kill'*, *feed'*, ... (each of which denotes some function in $\langle e, \langle e, t \rangle \rangle$)

Rule (to be revised directly): We number these *NER-n* (“Not-English Rule”) and since this rule is temporary it earns a *T*.

TNER-1. Given an expression α of the form $\langle [a], A/B, [[a]] \rangle$ and an expression β of the form $\langle [\beta], B, [[\beta]] \rangle$, there is an expression γ of the form $\langle [\alpha(\beta)], A, [[\alpha]]([[\beta]]) \rangle$.

(Note that we have introduced parentheses into the phonological string in order to make things easier to read. This may seem odd, but since there is no sense in which this language actually has a real phonology, it is harmless.) Thus, **TNER-1** combined with the lexicon above gives sentences like *squeak' (Porky')*, *chew-gum' (Dumbo')*, *feed' (Porky') (Dumbo')*, etc. It will also be useful to have rules introducing the logical connectives especially since these will be convenient when this language is used for writing meanings. We give the rules in phrase structure rule:

NER-2: $S_1 \rightarrow S_2 \ \& \ S_3$; $[[S_1]] = 1$ iff $[[S_2]] = 1$ and $[[S_3]] = 1$

NER-3: $S_1 \rightarrow S_2 \ \vee \ S_3$; $[[S_1]] = 0$ iff $[[S_2]] = 0$ and $[[S_3]] = 0$

NER-4: $S_1 \rightarrow S_2 \rightarrow S_3$; $[[S_1]] = 0$ iff $[[S_2]] = 1$ and $[[S_3]] = 0$

NER-5: $S_1 \rightarrow \sim S_2$; $[[S_1]] = 1$ iff $[[S_2]] = 0$

So far this is little more than a stripped-down version of the English fragment built earlier, with a strange word order. But the first main difference is that we now add in variables. So, in addition to *Fido'*, *Porky'*, *Dumbo'*, etc., the language also contains an infinite number of *NPs*, each of which is an expression of the form x_i for some positive integer *i*. In order to give the semantics, we introduce the notion of an *assignment function*—or, for short, an *assignment*. An assignment (function) is a function which takes each variable name and maps it to some object. The set of all

assignments will be called G , and g will be used to refer to an arbitrary assignment function. The semantics thus now has an extra dimension from what we have seen so far. A variable name does not have a semantic value in and of itself; it has a value *relative to some way to map it to some object*. In particular, for any g , the value of x_i on g is $g(x_i)$. Another way to say this is that the meaning of x_i is not an individual—it is a function from assignments to individuals (where the assignment itself is a function from variables to individuals). Thus another way to write what was said above is as follows: for any g , $[[x_i]](g) = g(x_i)$. As is the case with world and time parameters, the convention in much semantics literature is to write the g as a superscript, thus $[[x_i]]^g = g(x_i)$ (where there is a hidden “for all g ” in front of this formula).¹

The syntax now gives expressions like *sings'* (x_3). So far, we have an interpretation for the NP x_3 but not for the whole sentence. Exactly the technique used earlier for world and time arguments will be used here. Thus every expression in the language actually has a value only relative to an assignment. Put differently, the meaning of any expression is a function from the set of assignments to whatever you thought its type of meaning was before reading this chapter. NPs all have as their meanings functions from G to e . Ss denote functions from G to t , and so forth. But the semantics is constructed in such a way that many expressions denote constant functions from the set of assignments. This is true, for example, for all of the lexical items above except the variables. Thus $[[\text{Dumbo}]]$ is not d , but is a function

¹ This particular definition of an assignment introduces an odd status to a variable: it is both a syntactic expression and a member of a set which forms the domain of an actual function used in the semantics. If we wish the semantics to be concerned with model-theoretic objects only, then we have introduced variable names (a syntactic object) into our toolkit of model-theoretic objects; a rather unnatural situation. This, however, is just an artifact of the particular definition adopted above. One could revise this to take an assignment to be a function from the set of positive integers (clearly unobjectionable model-theoretic objects) to the set of individuals, such that for any g , $[[x_i]](g) = g(i)$. We will not adopt this convention in the remainder of the discussion because section 9.2 introduces variables over other types besides just individuals. We would then have to invoke fancier objects like ordered pairs of integers, ordered triples, etc. in the domain of the assignment functions; this would be needed to distinguish all of these different kinds of variables. All of that can be done but is unpleasant to read. We thus stick to the convention of having an assignment be a function from the variables themselves to other objects.

from G such that for all g in G , $[[\text{Dumbo}']](g) = d$. $[[\text{walks}']]$ is similar; it assigns every g in G some function in $\langle e, t \rangle$. Hence $[[\text{walks}']]$ is a function in $\langle G, \langle e, t \rangle \rangle$. (Of course once we return to English, whose expressions denote functions from world–time pairs to something else, we have one more layer in the functions: $[[\text{walks}]]$ is a function from worlds, times, and assignments to a function of type $\langle e, t \rangle$. It is constant on the assignment, but not on the world and time arguments. But the language here is purely extensional.)

The semantics of TNER-1 is now to be revised in the obvious way, and we now rename this as NER-1.

NER-1. Given an expression α of the form $\langle [a], A/B, [[a]] \rangle$ and an expression β of the form $\langle [\beta], B, [[\beta]] \rangle$, there is an expression γ of the form $\langle [a(\beta)], A$, where for all g , $[[\gamma]](g) = [[a]](g)([[\beta]](g))$. (Again, a more common notation for this last part is: $[[\gamma]]^g = [[a]]^g([[\beta]])^g$.)

Cumbersome though this may seem, it is well worth getting the intuition for what variables are doing here and how this aids in the compositional computation of a formula like *feed's* (x_1) (x_2) which contains two open variables. To illustrate, let us shrink the language to have only two variables x_1 and x_2 . Moreover, let us shrink the universe to only two individuals, p and f . Then there are four assignments as follows:

$$(1) \quad \begin{array}{llll} g_1: & x_1 \rightarrow p & g_2: & x_1 \rightarrow p \\ & x_2 \rightarrow p & & x_2 \rightarrow f \\ g_3: & x_1 \rightarrow f & g_4: & x_1 \rightarrow f \\ & x_2 \rightarrow p & & x_2 \rightarrow f \end{array}$$

Let $[[\text{feed}']]$ on each g be the function in (2):

$$(2) \quad p \rightarrow \begin{cases} p \rightarrow 1 \\ f \rightarrow 1 \end{cases} \quad f \rightarrow \begin{cases} p \rightarrow 0 \\ f \rightarrow 1 \end{cases}$$

We—the humans looking at this—can see that $[[\text{feed}' (x_1)(x_2)]]$ should come out false on g_3 and true on all the others. But that is because we can see the entire formula. We reason that for any g , $[[\text{feed}']](g)(f)(p)$ is false while $[[\text{feed}']](g)$ applied to any other combination of individuals is true. So we only need to look to see what assignment maps x_1 to f and x_2 to p , and that is g_3 . But the grammar is “dumb” and does not get to see the entire formula: it needs a procedure to compute the value in a fully compositional fashion. This is what is accomplished by the method of treating everything as a function from assignments to something else. To illustrate, the computation proceeds as follows:

$$(3) \quad \begin{array}{l} [[\text{feed}']] = g_1 \rightarrow \text{the above}; g_2 \rightarrow \text{the above, etc.} \\ [[x_1]] = g_1 \rightarrow p, g_2 \rightarrow p, g_3 \rightarrow f, g_4 \rightarrow f \end{array}$$

To derive the value of the next larger expression $feed(x_1)$ the compositional procedure has access only to the value of $feed'$ on each assignment and to value of x_1 on each assignment. By the rule NER-1, it computes the value in (4):

$$(4) \quad [[feed'(x_1)]] = g_1 \rightarrow \left\{ \begin{array}{l} p \rightarrow 1 \\ f \rightarrow 1 \end{array} \right\} \quad g_2 \rightarrow \left\{ \begin{array}{l} p \rightarrow 1 \\ f \rightarrow 1 \end{array} \right\} \quad g_3 \rightarrow \left\{ \begin{array}{l} p \rightarrow 0 \\ f \rightarrow 1 \end{array} \right\} \quad g_4 \rightarrow \left\{ \begin{array}{l} p \rightarrow 0 \\ f \rightarrow 1 \end{array} \right\}$$

The interesting point is the next step. To get from here to the value of the full formula, we could perfectly well erase the information in (2) and never look back at it. That information gives the value of $feed'$. But the compositional computation has no access to the meaning of smaller expressions like this, it “sees” only the value of the two expressions that combine to give the full sentence, and these are $feed'(x_1)$ and x_2 . The value of $feed'(x_1)$ is given in (4), and the value of x_2 is (5):

$$(5) \quad [[x_2]] = g_1 \rightarrow p, \quad g_2 \rightarrow f, \quad g_3 \rightarrow p, \quad g_4 \rightarrow f$$

These two pieces of information are sufficient to compute the right result for the full expression $feed'(x_1)(x_2)$ and this is shown in (6):

$$(6) \quad [[feed'(x_1)(x_2)]] = g_1 \rightarrow 1, \quad g_2 \rightarrow 1, \quad g_3 \rightarrow 0, \quad g_4 \rightarrow 1$$

The moral of this rather plodding exercise is to show that this technique allows the meanings to combine in a fully compositional fashion.

9.1.2. Quantifiers

In addition to their potential use in modeling natural language pronouns, variables are also used in first-order logic as a way to give a semantics for quantifiers like \forall (“for all”) and \exists (“there exists”). Recall that the goal in this chapter has nothing to do with a direct modeling of the English compositional syntax/semantics. The next additions we make to our Non-English language is very un-English-like; and an account of English quantification that respects the syntax will be the subject of the very next chapter. But working through the semantics for the first-order logic quantifiers again illuminates certain techniques (in particular, the notion of “binding” of the variables) that will be useful later. Moreover, having an explicit semantics for formulas with these quantifiers is useful in naming model-theoretic objects.

Hence we add the following two rules:

- NER-6. Given any expression α of the form $\langle [a], S, [[a]] \rangle$ there is an expression β of the form $\langle \exists x_i [a], S, \text{for all } g [[\beta]](g) = 1 \text{ iff there is some assignment } g' \text{ just like } g \text{ except possibly for the value assigned to } x_i \text{ such that } [[a]]^{g'} = 1 \rangle$.
- NER-7. Given any expression α of the form $\langle [a], S, [[a]] \rangle$ there is an expression β of the form $\langle \forall x_i [a], S, \text{for all } g [[\beta]](g) = 1 \text{ iff for all assignments } g' \text{ just like } g \text{ except possibly for the value assigned to } x_i [[a]]^{g'} = 1 \rangle$.

Syntactically, these rules prefix a quantifier and a variable to a sentence. (The brackets $[\dots]$ are used in two senses in the rules above. The outermost brackets in $\langle \exists x_i [a] \rangle$ are meant, as usual, to refer to the “phonology” of the expression—rather artificial here since this language is not pronounced—but nonetheless retained from the discussion of English to keep the differences minimal. The inner brackets around α are meant as actual parts of the string, just as parentheses were introduced in NER-1. The addition of these brackets ensures that each string is unambiguous.) NER-6 and NER-7 allow for any indexed variable to be introduced. Notice also that the rules are not formulated to require that there be an instance of x_i within the sentence to which the quantifier and variable are prefixed. There is no particular reason to complicate the rule to require such a variable within the S ; if there is none, the effect of these rules on the semantics will just be vacuous—the meaning of β is simply the same as the meaning of α . (This is thus referred to as *vacuous quantification*.)

Let us walk through the semantics. NER-6 ensures that a sentence of the form $\exists x_i S$ is true on an assignment g if there is some g' which differs only on the value assigned to i such that the lower S is true on g' . What does it take to find such an assignment? Informally it requires there to be some individual a who can be slotted in to the position held by x_i which would make the sentence true. In other words, if the sentence were $\exists x_i [\text{walks}'(x_i)]$ and we were evaluating this on g , we would look to find some individual a such that $\text{walks}'(x_i)$ is true on the g' where x_i is mapped to a . In turn, this means that there has to be some individual who did indeed walk. Note that the phrasing “ g' just like g except possibly for the value assigned to x_i ” means that g' could of course be g itself. For the case of universal quantification, the semantics ensures that $\text{walks}'(x_i)$ is true for every assignment which differs from g only in the value that it assigns to x_i . Suppose for example that there are 23 individuals. Then there are 23 assignments that vary only on the value assigned to x_i (for any i). (One of

these is guaranteed to be g itself.) Since this must be true on all of these, it is ensured that every individual does indeed walk.

***9.1.** Show that application of NER-6 and NER-7 in cases where the input sentence does not contain an instance of the variable introduced in these rules has no semantic effect. (In other words, show that vacuous quantification is indeed vacuous.)

There is another way to phrase the semantics which is equivalent. First we introduce a notational convention. For any g and any individual a , let $g^{x_i \rightarrow a}$ mean the assignment just like g except that x_i is assigned the value a (where of course this again could be g itself). Then the semantic parts of NER-6 and NER-7 could be rewritten as follows:

NER-6'. ... for all g , $[[\beta]](g) = 1$ iff there is some individual a such that $[[\alpha]](g^{x_i \rightarrow a}) = 1$.

NER-7'. ... for all g , $[[\beta]](g) = 1$ iff for every individual a , $[[\alpha]](g^{x_i \rightarrow a}) = 1$.

These amount to the same thing as given above. If there are n individuals, then there are n assignments just like g except possibly for the value assigned to x_i . Take the set of all assignments which differ from g only on (possibly) the value assigned to x_i and call that set g/x_i . (A caution on the notation: $g^{x_i \rightarrow a}$ is a single assignment; g/x_i is a set of assignments.) There is a one-to-one correspondence between the set of individuals and the set g/x_i . NER-6' ensures that $[[\beta]]$ will be true on an assignment g if we can find an individual a such that $[[\alpha]]$ is true on $g^{x_i \rightarrow a}$. But if that is the case, then it of course follows that there is a g' in g/x_i such that $[[\alpha]]$ is true on g' , and this is what NER-6 asks for. Conversely, if there is such a g' , then it follows that there is some individual a such that $[[\alpha]]$ is true on $g^{x_i \rightarrow a}$. Similar reasoning holds for NER-7 and NER-7'. The key is in noticing that there is a one-to-one correspondence between the set of individuals and the set g/x_i . Some terminology will be useful below. Consider a set A , and imagine "slicing it up" into a set of subsets such that each member of A occurs in one and only one of these subsets. This is called a *partition* of A , and each subset is called a *cell* in the partition (see the Appendix to Chapter 1). We will use the term *cellmates* to refer to members of A which are in the same cell. The relevance of this here is that each variable x_i induces a partition on G (the set of all assignments); each cell in the partition is the set of assignments that differ

only on the value assigned to x_i . As noted above by one example, the number of members in each cell will be exactly the number of individuals (and each cell has the same number of assignments in it). Thus g/x_i defined above is a cell in the partition induced by x_i . We will see this played out by example below, but we can note here that the semantics of NER-6 and NER-7 are such that all members of the cell of a partition defined by the variable introduced by the relevant rule will receive the same value.

By way of concrete illustration, consider the composition of the two sentences in (7), given the world set out earlier (two individuals: $\{p, f\}$ and $[[\text{feed}']]$ as shown in (2) and repeated here):

- (7) a. $\exists x_1[\forall x_2 [\text{feed}'(x_1)(x_2)]]$
 b. $\forall x_1[\forall x_2 [\text{feed}'(x_1)(x_2)]]$
- (2) $p \rightarrow \begin{cases} p \rightarrow 1 \\ f \rightarrow 1 \end{cases} \quad f \rightarrow \begin{cases} p \rightarrow 0 \\ f \rightarrow 1 \end{cases}$

We—the all-seeing human beings—know that (7a) should come out true regardless of the assignments (there is indeed someone that everyone fed, namely p) while (7b) should come out false (it is not the case that everyone was fed by everyone; p was fed only by p). But recall again that the compositional computation is not smart and has no access to the meanings of deeply embedded expressions (such as *feed'*). In each step, this computation has access only to the semantics of the input expressions, and—having added in NER-6 and NER-7—it also has access to the structure of the assignments. We will take the assignments as given in (1) (using again only two variables):

- (1) $g_1: \begin{matrix} x_1 \rightarrow p \\ x_2 \rightarrow p \end{matrix} \quad g_2: \begin{matrix} x_1 \rightarrow p \\ x_2 \rightarrow f \end{matrix} \quad g_3: \begin{matrix} x_1 \rightarrow f \\ x_2 \rightarrow p \end{matrix} \quad g_4: \begin{matrix} x_1 \rightarrow f \\ x_2 \rightarrow f \end{matrix}$

We already showed the compositional computation of $\text{feed}'(x_1)(x_2)$, the final step in that was (6), repeated here:

- (6) $[[\text{feed}'(x_1)(x_2)]] = g_1 \rightarrow 1, \quad g_2 \rightarrow 1, \quad g_3 \rightarrow 0, \quad g_4 \rightarrow 1$

Looking at (7a), the next step is to compute $[[\forall x_2 [\text{feed}'(x_1)(x_2)]]]$ on each g . Here we need access to the structure of the assignments. Computing this for g_1 , we look to at its value on all g' just like g_1 except for the value assigned to x_2 . In other words, we look to see if it is true for all members of g/x_2 . That set is g_1 and g_2 . (The partition on G induced by x_2 is $\{\{g_1, g_2\}, \{g_3, g_4\}\}$.) Looking only at (6), we see that the value of $[[[\text{feed}'(x_1)(x_2)]]]$ is 1 on both g_1

and g_2 . Hence, the value of the larger sentence $\forall x_2 [\text{feed}'(x_1)(x_2)]$ maps both g_1 and g_2 , to 1. Turning now to the value of the larger sentence on g_3 and g_4 , they too form a single cell, and each will receive 1 if and only if the expression in (6) assigns 1 to both assignments. It does not; it assigns 0 to g_3 . From this we derive the following:

$$(8) \quad [[\forall x_2 [\text{feed}'(x_1)(x_2)]]] = g_1 \rightarrow 1, \quad g_2 \rightarrow 1, \quad g_3 \rightarrow 0, \quad g_4 \rightarrow 0$$

The next step is to compute $[[\exists x_1 [\forall x_2 [\text{feed}'(x_1)(x_2)]]]$ on each g . Consider g_1 . Here the relevant set of assignments g/x_1 is the set $\{g_1, g_3\}$ so those two assignments are guaranteed to have the same value. g_1 will of course get the value 1. But so will g_3 even though the line just above assigned 0 to g_3 . At this step, however, g_3 no longer cares only about itself—it merely needs to find some cellmate in the partition induced by x_1 that is assigned 1, and g_1 fits the bill. The same will apply to the computation on g_2 and g_4 . They are cellmates and hence look to each other; g_2 is assigned 1 by the value of the inner S shown in (8), and so both g_2 and g_4 will be assigned 1. The sentence thus comes out true on all assignments: it is technically a function from assignments to truth values but is a constant function (and so we tend not to detect the assignments, and don't think of it as assignment-dependent).

9.2. Show the last line of the computation of (7b).

Notice that at each step shrinks the space of possible ways the assignments can diverge in their value. At the step of interpreting $\forall x_2 [\text{feed}'(x_1)(x_2)]$, we have an equivalence class that partitions the assignments into $\{\{g_1, g_2\}, \{g_3, g_4\}\}$. At the next stage we have the partition $\{\{g_1, g_2\}, \{g_3, g_4\}\}$. By the end we are guaranteed a constant function: g_1 will have to have the same value as g_2 by the first step, but will have to have the same value as g_3 by the second step. And g_3 and g_4 must have the same value by the first step. So at the end, all four have the same value. Incidentally, we can now give a (rough) semantic definition of what it means for a variable to be “bound”: x_i is bound within an expression if all members of g/x_i receive the same value. Put differently, it doesn't matter what value one chooses for x_i —the set of assignments in g/x_i in fact differ on just that—but because they are all in the same cell and look to each other, the fact that they disagree on the value of x_i is irrelevant.

Otherwise x_i is free (and so the value of the expression containing it still depends on what value is assigned to x_i). We say “rough” here simply because there are cases that correspond to the intuitive notion of a variable being free but where all assignments are nonetheless assigned the same value (even if we were to fold in worlds) as, for example, the case of a tautology. Thus a statement like x_3 is x_3 (given a suitable definition for *be* here) is assigned the same value (1) on all assignments and in all worlds, but—should we really care about a precise definition of a “bound variable”—we would probably want to take x_3 to be “free” here. But this is a problem only if we really ever need a precise definition.² As we will see later, this is a notion for convenience only. If the grammar never needs to refer to a “bound variable” (or, in the case of English, a “bound pronoun”) then our loose definition is perfectly adequate since it is intended just for convenience in discussion. Indeed, the fragment to be developed for the rest of this book has no need for the grammar to refer to a notion of “binding.”³ And this is equally true for the version that makes use of assignment functions to model pronoun meanings as for the variable-free version.

² Heim and Kratzer (1998) develop a very interesting system using partial functions. Their system is meant to model English (rather than the artificial toy language here) but the same points would hold. Thus each expression has a value relative to some assignment, but the assignments themselves can be partial functions from the set of variable names to individuals. Moreover, the value of some sentence S is defined only for a subset of assignments (only for those whose domain is the set of variables which occur in what we intuitively think of as “unbound” within S). Space precludes discussion of this here, but it does allow for a definition of “bound” vs “free” variables (or pronouns) that avoids the problem noted above. However, as mentioned above, there will turn out to be no need for a formal semantic definition of “bound” vs “unbound” pronouns (or variables).

³ There is much work within the linguistics literature that uses the term “binding” to state various constraints relevant to the distribution and readings of pronouns and reflexives. The goal here under Direct Compositionality is to have the empirical generalizations that are packed into constraints on “binding” follow from the way the system is set up without the grammar needing any such definition. This book will not have space to discuss so-called binding constraints, but see Chapter 15 for a discussion of what is known as Weak Crossover effects.

9.3. Take a universe with three individuals $\{p, f, k\}$. Take the language given above but with just the two variables x_1 and x_2 . We are concerned here with just one world, which is such that on any assignment g , $[[kiss]](g)$ is:

$$p \rightarrow \begin{cases} p \rightarrow 1 \\ f \rightarrow 0 \\ k \rightarrow 1 \end{cases} \quad f \rightarrow \begin{cases} p \rightarrow 1 \\ f \rightarrow 1 \\ k \rightarrow 0 \end{cases} \quad k \rightarrow \begin{cases} p \rightarrow 0 \\ f \rightarrow 1 \\ k \rightarrow 0 \end{cases}$$

- (a) First, show the nine assignments.
 (b) Then show the compositional computation for each of the following, showing each step as a function from the domain of assignments to some other value:
 (i) $\forall x_1 [\exists x_2 [kiss'(x_2)(x_1)]]$
 (ii) $\exists x_2 [\forall x_1 [kiss'(x_2)(x_1)]]$
 (iii) $\forall x_2 [kiss'(x_2)(x_2)]$

Note that once you have shown the computation of $kiss'(x_2)(x_1)$ for (i) there is no reason to repeat this for (ii); just recycle what you have done.

- (c) Consider (iv):
 (iv) $\forall x_1 [kiss'(x_1)(x_1)]$

The part within the brackets will not be the same as the corresponding part for (iii). But (iv) (regardless of the structure of the world in question) is guaranteed to have the same value as (iii). State why this is so.

9.2. The lambda calculus

9.2.1. Lambda abstraction

Continuing with the digression into languages which are not English, we now take the language above as the point of departure and enrich it, developing what is known as the lambda calculus. As noted earlier, this gives us a precise way to name complex model-theoretic objects. Note, though that developing this language has no theoretical status in that regard—one could continue to use prose to name the object. However, our prose has been getting rather clumsy—and would get even more so more complex objects are developed.

The first addition to the language is an infinite set of variables over any kind of object (and with every syntactic category). There are variables whose category is S/NP and semantic type $\langle e, t \rangle$; we will name these P_1 , P_2 , etc. There are variables over propositions (category S) which we will

label p_1, p_2 , etc., and variables of type $\langle e, \langle e, t \rangle \rangle$ (category (S/NP)/NP) which we will label R_1, R_2 , etc. Recall that, for convenience, we took the domain of the assignments to be the variable names themselves, giving variable names a kind of double status (syntactic objects, and objects in the domain of assignments) and we continue with this convention here. We add the rule in NER-8; X and Y are variables over any category, and y semantic type of expressions of category Y (for any g and any expression β of category Y , $[[\beta]](g)$ is a member of y).

NER-8. If α is an expression of the form $\langle [a], X, [[a]] \rangle$ and v is a variable of the form $\langle [v], Y, [[v]] \rangle$ then there is an expression β of the form $\langle [\lambda v [a]], X/Y, \text{for any } g, [[\beta]](g) \text{ is a function from the set of objects of type } y, \text{ such that for each } a \text{ in } y, [[\beta]](g)(a) = [[a]](g^v \rightarrow a) \rangle$.

As was the case with NER-6 and NER-7, we are using square brackets in two ways in the “phonology” part of NER-8. The outer brackets indicate “the phonology of” the new expression; the inner brackets around α are actually introduced by the rule (to keep the strings from being ambiguous).

Let us walk through the semantics part in prose. First, let x be the semantic type of expressions of category X and y the type of expressions of category Y . Then our new expression has some meaning which is a function from assignments to something in $\langle y, x \rangle$. On any assignment g , this new expression takes every member a of the y set, and assigns to it the value that the inner formula (α) assigned to the assignment just like g except where a is the value of the variable v . Prefixing a λ and a variable v to some formula is called *lambda-abstraction* over that variable, but the important point is to see its semantic effects. One effect is it takes a formula which is non-constant on g/v and turns it to a formula whose value is the same for all members of g/v . The other effect is that it takes an expression which, on any assignment, has as its value something in set a , and returns an expression which (on any assignment) is a function from some set b to a . Hence the expression α to which is being prefixed λv has as its value a function in $\langle G, a \rangle$. Let the variable v be of type b , then the full expression $\lambda v [a]$ has as its value a function in $\langle G, \langle b, a \rangle \rangle$.

By way of example, take a universe with two individuals $\{p, f\}$ and put just two individual variables x_1 and x_2 in our language. The four assignments are as given earlier in (1):

$$(1) \quad \begin{array}{llll} g_1: & x_1 \rightarrow p & g_2: & x_1 \rightarrow p & g_3: & x_1 \rightarrow f & g_4: & x_1 \rightarrow f \\ & x_2 \rightarrow p & & x_2 \rightarrow f & & x_2 \rightarrow p & & x_2 \rightarrow f \end{array}$$

Suppose that we are considering a world in which $[[squeaks']]$ for all assignments is $p \rightarrow 1$ and $f \rightarrow 0$. Let us compute the value of $\lambda x_1[squeaks'(x_1)]$. First, we compute $[[squeaks'(x_1)]]$ on each g and this gives:

$$(9) \quad g_1 \rightarrow 1, \quad g_2 \rightarrow 1, \quad g_3 \rightarrow 0, \quad g_4 \rightarrow 0$$

The next step is to compute the value of the whole formula. Once again, the compositional procedure does not need to consult anything about the inner formula; the computation computes the value of $\lambda x_1[squeaks'(x_1)]$ on each assignment purely by looking at (9) and at the structure of the assignments. In particular, the variable that is “bound” here by the λ is x_1 ; the relevant partition is thus $\{\{g_1, g_3\}, \{g_2, g_4\}\}$. It is guaranteed that for any given cell, each member in that cell will receive the same value; this is because they “look to each other.” For example, consider the computation of the value of $\lambda x_1[squeaks'(x_1)]$ on g_1 . First, note that the value of this on g_1 is going to be a function of type $\langle e, t \rangle$, assigning 1 or 0 to p and 1 or 0 to f . The determination of what value is assigned to p is made by looking at the value of the inner formula on the assignment just like g_1 except that x_1 is mapped to p . This is g_1 , so (8) tells us that p will map to 1. For f , the relevant assignment is g_3 so (9) tells us that f maps to 0. Exactly the same remarks hold when we want to find the value of $\lambda x_1[squeaks'(x_1)]$ for g_3 . Since g_1 and g_3 are in the same cell, they are guaranteed to be assigned the same value by the full formula. The same holds for g_2 and g_4 . But since there is no x_2 at issue, these will come out the same as g_1 and g_3 , and in the end we have the constant function from assignments: on any g the value of $\lambda x_1[squeaks'(x_1)]$ is $p \rightarrow 1$ and $f \rightarrow 0$.

The reader might have noticed that the value of $\lambda x_1[squeaks'(x_1)]$ is exactly the same as the value of $squeaks'$, and was destined to be so. By rethinking the lambda expressions as ways to name functions (which is ultimately the way we will be using them), this is not surprising: the full formula is (on any assignment) a function which asks for an individual, and gives the value of $squeaks'$ applied to that individual. So of course this is just a fancy way to write (semantically) exactly the same object as $squeaks'$. Note that of course $\lambda x_2[squeaks'(x_2)]$ is also the same object. As a second example, consider the formula in (10):

$$(10) \quad \lambda x_1[\lambda x_2[feed'(x_1)(x_2)]]$$

The semantics of this is exactly the same as the semantics of $feed'$. Again, thinking of these as ways to name functions, one can convince oneself of this

by noting that for any assignment g , this is a function that takes one individual and then another and returns the value that *feed'* returns when applied to the first individual and then the second. And that, of course, is exactly the *feed'* function (a function of type $\langle e, \langle e, t \rangle \rangle$).

9.4. Given $[[feed']]$ as shown in (2) (and continuing with a language with only two variables and a universe with only the two individuals $\{p, f\}$), demonstrate the truth of the remarks above by showing the full compositional computation of (10). Remember that the end product should be a function from assignments to functions of type $\langle e, \langle e, t \rangle \rangle$.

9.5. Like the quantification rules, NER-8 does not require there to be an instance of the variable introduced by this rule within α . If there is none, the rule is not vacuous as in the case of the quantification rule, as the meaning of β is not even the same type as the meaning of α . State what the meaning of β is in general if there is no occurrence of the relevant variable within α .

In view of these equivalences one might wonder what the point is of this notation. It is because other kinds of values can be written in this notation. Take for example (11):

(11) $\lambda x_1 [\lambda x_2 [\text{kiss}'(x_2)(x_1)]]$

This does not have the same meaning as any other lexical item in the language (keeping our lexicon to be just a subset of English), but corresponds to something that we can roughly paraphrase as *was-kissed-by'*. We return to this below.

9.2.2. Lambda conversion

Since the ultimate interest in the lambda calculus is as a way to name various functions, it is also useful to understand ways in which one can simplify various formulas in the lambda calculus. By “simplify” we mean naming the same object in a shorter way. We can actually give proofs of the equivalences that we show here, but to save space and possible tedium, we will try only to give the intuition behind why the equivalences hold.

Take any formula of the form $\lambda u[a](c)$. Call $a/u \rightarrow c$ the formula derived from $\lambda u[a](c)$ except where all occurrences of u in a are replaced by c . In general (but not always—the exception to be discussed directly) these two formulas will always have the same value. This is called *lambda conversion*. We have already seen a few instances of this. Thus (12a) and (12b) are guaranteed to have the same value; note that (12b) is the result of applying lambda conversion to (12a):

$$(12) \quad a. \lambda x_1[\text{squeak}'(x_1)](\text{Porky}') = b. \text{squeak}'(\text{Porky}')$$

The sign “=” here means *has the same value* (on, of course, all assignments). Another example is (13):

$$(13) \quad a. \lambda x_1[\lambda x_2[\text{feed}'(x_1)(x_2)]](\text{Porky}') = b. \lambda x_2[\text{feed}'(\text{Porky}')(x_2)]$$

We also know from the sort of equivalences discussed in 9.2.1 that (b) is equivalent to $\text{feed}'(\text{Porky}')$. Similarly, the pair in (14) are equivalent:

$$(14) \quad a. \lambda x_1[\lambda x_2[\text{feed}'(x_2)(x_1)]](\text{Porky}') = b. \lambda x_2[\text{feed}'(x_2)(\text{Porky}')]$$

Informally, both of these (on all assignments) denote the set of individuals fed by Porky. We can show cases of two lambda conversions, as in (15):

$$(15) \quad \begin{aligned} a. & \lambda x_1[\lambda x_2[\text{feed}'(x_2)(x_1)]](\text{Porky}')(\text{Dumbo}') = \\ b. & \lambda x_2[\text{feed}'(x_2)(\text{Porky}')](\text{Dumbo}') = \\ c. & \text{feed}'(\text{Dumbo}')(\text{Porky}') \end{aligned}$$

(In doing lambda conversions, one needs to make sure to pay attention to the brackets. In (15a) Porky' must convert in for x_1 because it is the argument of the function shown in (11).)

9.6. Simplify the following by lambda conversion:

$$(i) \lambda x_1[\lambda x_2[\text{feed}'(x_2)(x_1)]](\text{Porky}')(\text{Dumbo}')$$

However, it is not always the case that $\lambda u[a](c)$ is equivalent to $a/u \rightarrow c$. Rather than giving a full explanation of when and why the equivalence does not hold, we will simply give a recipe for determining when lambda conversion is not valid (i.e., when it is not guaranteed to result in an equivalent formula) along with an example to help elucidate why. This will be followed by a brute-force recipe for allowing conversion on a slightly different formula.

Hence we first give a purely syntactic definition of the notion of a “bound” variable. (Recall that earlier we gave a rough semantic definition; a variable x_i is bound in a formula if the value of that formula is the same on all g/x_i . But for the purposes of the recipe, it is easiest and most practical to see the point with a syntactic definition of the notion bound. Leaving aside contradictions and tautologies, the two definitions converge on the same result.) Hence we will define *bound* as follows:

(16) Let O (for *operator*) stand for any of λ , \exists , and \forall .

Given any formula of the form $Ox[\dots]$, anything within the brackets is *within the scope of* O (for x a variable).

A variable x with the name v_i is *bound* by an operator O iff:

- (i) the variable next to O in the formula is v_i ;
- (ii) x is in the scope of O ; and
- (iii) there is no other O' whose adjacent variable is v_i such that x is in the scope of O' and O' is in the scope of O .

Notice that this definition allows for variables which are free within some subformula even though they may be bound within the main formula. For example, x_1 is bound within the formula $\lambda x_1[squeaks'(x_1)]$ but is free within the subformula $squeaks'(x_1)$. Then whenever there is a formula of the form $\lambda u[a](c)$ one cannot perform lambda conversion (and be guaranteed to preserve equivalence) just in case c is or contains a variable which has the same name as one which is bound within a . Intuitively, this is because the variable c or the variable within c will be accidentally “caught” by the operator inside a .

As an example, take the pair in (17), where (17b) is the result of the illegitimate lambda conversion:

(17) a. $\lambda x_1[\exists x_2[\text{feeds}'(x_1)(x_2)]](x_2)$ b. $\exists x_2[\text{feeds}'(x_2)(x_2)]$

The truth of (17a) is assignment-dependent—it depends on the value assigned to x_2 . Paraphrasing it in prose, it says that x_2 is such that there is someone who feeds it. This is because the lambda expression within (17a) is the (assignment-dependent) function that maps an individual to true just in case there is someone who feeds it. The whole formula then says that this function truly applies to x_2 , but whether or not that is true depends on the value we assign to x_2 . (17b), on the other hand, is constant on all assignments—on any g , it simply asserts that there is someone who feeds themselves. What has gone wrong is that the x_2 bound by the existential quantifier in (17a) and the x_2 outside the lambda expression are not really

the same. The bound one inside the lambda expression could as well have been x_1 , x_3 , or anything else: it is bound so its name no longer matters. The one outside the expression is not bound, so its name does matter. Put differently, $\lambda x_1[\exists x_2[\text{feeds}'(x_1)(x_2)]]$ has the same value on all g/x_2 . But the outermost x_2 does not have the same value on all g/x_2 , and so neither does the full formula in (17a).

There is also a simple way to avoid mistaken lambda conversions such as those shown in (17). Simply find some new variable that appears nowhere else, take all of the *bound* occurrences of the colliding variable inside α (including the occurrence that sits next to the operator that binds it), and rename it with the new variable. One should be able to convince oneself that once a variable is bound its name no longer matters, so renaming it in this way will have no effect on the meaning of α . Then the lambda conversion can proceed. Given any expression α with a variable a bound within it, call α' the new expression in which every occurrence of a is replaced by another variable b . α and α' are said to be *alphabetic variants*, which means that they are guaranteed to have the same value on all assignments.

***9.7.** Show that one cannot instead rename the free variable within the expression (or which is the expression) which is being converted in. You can show this by example and/or discuss more generally why the equivalence is not guaranteed to hold in if one renames the free variable.

9.8. Take each of the following formulas, and simplify them as much as possible (i.e., write them in a shorter form that preserves their meaning). Recall that P_1 is a variable of type $\langle e, t \rangle$ and R_1 is a variable of type $\langle e, \langle e, t \rangle \rangle$.

- (i) $\lambda x_1[\text{kiss}'(x_1)(x_2)](x_2)$
- (ii) $\lambda x_1[\text{kiss}'(x_1)(x_2)](x_1)$
- (iii) $\lambda P_1[\forall x_1[P_1(x_1)]](\text{feeds}'(x_1))$
- (iv) $\exists x_2[\lambda R_1[R_1(x_2)(x_2)]](\lambda x_3[\lambda x_1[\text{grooms}'(x_3)(x_1)]])$
- (v) $\lambda P_1[\lambda x_1[P_1(x_1)]](\text{chews-gum}')$

9.3. Naming model-theoretic objects

The advantage of having the explicit notation above—along with the principles for rewriting lambda expressions as simpler (syntactically) but equivalent (semantically) expressions—is that it provides a tool to write out complex functions which are difficult to state in prose. It also allows a much less clumsy semantics for some of the earlier rules. Notice that although the meanings of lambda formulas are stated with reference to assignments and variables, we can remain for the moment noncommittal as to the status of these for natural language semantics and still use the relevant formulas. The reason is simple but quite important: for now, at least, *all of the model-theoretic objects that we will name using the lambda notation contain no unbound variables within the formulas*. Thus if assignments were an actual part of the machinery, all of the objects of concern at this point are *constant* functions from the set of assignments. But if everything is a constant function from the set of assignments, we can just as well do without the assignments. Thus the use of assignments, variables, lambda conversion, and so forth is important to understand to make sure that we do correctly represent the meanings of the natural language expressions. But these tools are (so far) not part of the linguistic semantic system. Whether or not something like variables and lambda abstraction do play a role in the semantic theory is addressed in Part III.

To give a few examples of the usefulness of this notation for writing meanings, we will slightly modify the conventions above. First, let us use *feed'* not just as a word in the non-English language, but let us also use it to represent the meaning of English *feed*. In other words, it is the same as $[[\text{feed}]]$. The reason for having both ways to represent the meaning of *feed* has to do with the fact that each notation has certain advantages in terms of readability, and so we will alternate back and forth between the two notations as convenient. Second, the convention of numbered variables is sometimes difficult to read. We will thus use x, y, z , etc. for variables of type e ; P and Q for variables of type $\langle e, t \rangle$; p and q to be variables of type t , and others as needed. When introducing variables whose type is not clear we will subscript them, as for example in the following:

$$(18) \quad \lambda R_{\langle e, \langle e, t \rangle \rangle} [R(\text{Fido}')(\text{Dumbo}')]]$$

This expression names a function in $\langle \langle e, \langle e, t \rangle \rangle, t \rangle$: it characterizes a set of two-place relations. In particular, it maps a relation R to true just in case

$R(\text{Fido})(\text{Dumbo})$ is true or, informally, just in case “Dumbo Rs Fido.” Or, take the following (here we omit the subscript, keeping R to be of the same type as in (19)):

(19) $\lambda R[R(\text{Porky})]$

This is a function of type $\langle\langle e, \langle e, t \rangle \rangle, \langle e, t \rangle\rangle$: it maps any two-place relation R to a function that characterizes the set of individuals who “ R Porky.” If this function were applied feed’ (or, $[[\text{feed}]]$) it would yield the set of Porky-feeders.

We can write out the meanings that we gave above for *to* and for *be*. The former, being the identity function on individuals, is (20a) and the latter is (20b):

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

Or, consider the account of passive in English according to which (21) is not derived by any kind of movement rule, but where there is a productive (morphological) rule mapping an active verb into a passive verb, changing its syntax (and phonology) and its meaning:

(21) The squirrel was chased by Mitka.

We will not formalize the rule here (the reader can easily do so), but we can note that its output would be the verb *killed* with a category roughly $(S[\text{Part}]/NP)/PP[\text{BY}]$ (we leave it open as to the best feature specification here on the result) and with the meaning shown in (22a) or—just using a slightly different notation—(22b) (whose visual awkwardness reveals the rationale for the prime notation):

(22) a. $\lambda x[\lambda y[\text{chase}'(y)(x)]]$ b. $\lambda x[\lambda y[[[\text{chase}]](y)(x)]]$

Note that the rules for λ -conversion are now quite useful as we build up the compositional computation for (21).

9.9. To show the full computation of (21) requires a treatment of the auxiliary. Pretend, then, that the passive verb is actually the single verb *was-chased* with category $(S/NP)/PP[\text{BY}]$ and meaning as given above. With that, show the full syntactic and semantic computation of (21), doing λ -conversions wherever (and as soon as) you can to simplify the way in which the semantics is stated. (Note: It is always helpful in terms of the exposition to do the λ -conversions as soon as possible.)

***9.10.** Assume the Wrap analysis of three-place verbs discussed in Chapter 5 and explicitly folded into the fragment in Chapter 6. That is, assume that all verbs of the form $((S/NP)/NP)/X$ (for X any category) are

marked as $((S/NP)/_wNP)/X$. (This is actually not crucial for the main point of this question, but since the implementation of the question requires some decision about the structure of three-place verbs, we will make that assumption.)

Since earliest Transformational Grammar, it has been noted that there are a large number of verbs that occur both in the V-NP-PP[TO] configuration and in the V-NP-NP configuration, and where sentences of the form in (ia) have the same truth conditions as those of the form (ib):

- (i) a. NP_1 V NP_2 to NP_3 b. NP_1 V NP_3 NP_2

Examples of this are shown in (ii)–(iii) and this is generally referred to as the *dative alternation*:

- (ii) a. Sabrina gave the toy to Fido.
b. Sabrina gave Fido the toy.
- (iii) a. George told the Cinderella story to the children.
b. George told the children the Cinderella story.

Many studies of this (beginning with Oehrle 1976) treat these alternations as a unary rule mapping words to others. Take the verbs in the (a) form to be basic. Formulate the rule in question, showing how the output expression differs from the input in syntactic category and in meaning. (The point, of course, is that the use of λ s makes it easier to state the meaning of the output verb in terms of the meaning of the input verb. You do not need to use any λ s to state the meaning of the input verb; the meaning of the lexical item *give* in (iia), for example, is simply $[[give]]$.)

An interesting sidelight: It is often claimed that the rule is not really productive and regular, for there are many verbs that do occur in the (a) configuration but not in the (b) configuration; examples include *contribute*, *recount*, *donate*, etc. These contrast with *hand*, *toss*, *tell*, *give*, and many others which do occur in both configurations. Leaving aside the verb *promise*, state the generalization of the verbs that do undergo the rule in phonological terms, and think about how one can state the rule so that it is indeed a productive unary rule.

One final example is the Curry'ed meaning for *and* (when it combines two expressions whose meanings characterize sets), which was given earlier in terms of set intersection. We can equivalently give its meaning as follows, where a is meant to be any type:

$$(23) \quad \lambda A_{\langle a, t \rangle} [\lambda B_{\langle a, g \rangle} [\lambda z_a [A(z) \ \& \ B(z)]]]$$

Finally, this notation also allows us to state the semantics of many of the earlier rules in a much more elegant fashion. R-3 (from Chapter 8) serves to illustrate this, but first one more piece of notation. Given an individual x and a function P of type $\langle e, t \rangle$, we will use the notation $\iota x[P(x)]$ to mean the unique (or most) contextually salient individual such that $P(x)$ is true. (Often the iota operator shown here is used to mean the unique such individual; here we use it in the same way that English *the* is used.) Note then that any expression of the form $\iota x[a]$ denotes an individual. With this notation, we restate R-3:

R-3. If α is an expression of the form $\langle [a], N/PP[OF], [[a]] \rangle$ then there is an expression β of the form $\langle [\beta], NP/_L NP[GEN], \lambda x \iota y [[[a]](x)(y)] \rangle$.

9.11. Use the λ -notation to restate R-2 from Chapter 8.

9.12. Give the meaning of *didn't* in λ -notation.