Chapter 7
Lambda Abstraction

In this chapter we are going to study the λ-operator in simplified form. The symbol λ is the Greek letter lambda. We will first consider the logic of the λ-operator abstractly by introducing it in the intensional predicate calculus (IPC) presented in chapter 5. We will subsequently discuss several of its applications to natural language semantics. Such applications will illustrate the usefulness of the λ-operator in semantics.

1 An Elementary Introduction to Lambda Abstraction

For our present purposes we can regard the λ-operator as a way of systematically defining complex properties in terms of properties already given. Let us see how it works by introducing it in IPC. Add to IPC the following syntactic rule.

(1) If ψ is a well-formed formula and x a variable, $\lambda x[\psi]$ is a Pred_1.

The expression $\lambda x[\psi]$ should be read as "the property of being an x such that ψ." We say that x in $\lambda x[\psi]$ is bound by λ and that $[\psi]$ is the scope of that occurrence of the λ-operator. We will sometimes take the liberty of omitting the square brackets in $\lambda x[\psi]$ (we will instead write $\lambda x\psi$) if the context makes it clear what the scope of the λ-operator is.

Since an expression of the form $\lambda x[\psi]$ is a one-place predicate, we can apply it to terms and obtain well-formed formulas. In (2) we give some examples of λ-expressions and in (3) some well-formed formulas derived from them.

(2) *a.* $\lambda x[\neg\mathrm{married}(x) \wedge \mathrm{male}(x) \wedge \mathrm{adult}(x)]$
b. $\lambda x \exists y[\mathrm{love}(y, x)]$

(3) *a.* $\lambda x[\neg\mathrm{married}(x) \wedge \mathrm{male}(x) \wedge \mathrm{adult}(x)](j)$
b. $\lambda x \exists y[\mathrm{love}(y, x)](j)$

If we have in the language of IPC the predicates *married*, *male*, and *adult*, then (2*a*) illustrates how we can define the property of satisfying simultaneously the property of not being married, of being male, and of being an adult. The property defined in (2*a*) can be thought of as the property of being a bachelor. By the same token, if we have in our language the *love* relation, then we can define the property of being loved by someone (the property of being an x such that for some y, y loves x), as illustrated in (2*b*).

These properties can be predicated of individuals. So (3*a*) says that John has the property of being an x such that x is unmarried, x is male, and x is an adult. Under what conditions do we want to say that John has such a property? If and only if he is not married, is male, and is an adult. Similarly, under what conditions will John have the property of being loved by someone? Just in case there is someone who loves John. This means that we want (3*a*, *b*) to have the same truth conditions as (4*a*, *b*), respectively.

(4) *a.* $[\neg\mathrm{married}(j) \wedge \mathrm{male}(j) \wedge \mathrm{adult}\,(j)]$
b. $\exists y[\mathrm{love}(y, j)]$

The syntactic relations between (3*a*, *b*) and (4*a*, *b*) are fairly obvious. Formulas (4*a*, *b*) are obtained from (3*a*, *b*) by dropping the λx at the beginning, dropping the (j) at the end, and replacing x with j in the body of the λ-expression. This is the syntactic realization of the semantic relation that we want in general to hold between $\lambda x\psi(t)$ and ψ. We can schematize this as in (5).

(5) $\lambda x[\psi](t) \leftrightarrow \psi[t/x]$, where t is any term (an individual variable or constant) and $\psi[t/x]$ is the result of substituting t for all occurrences of x in ψ.

Rule (5) is a simple generalization of what we did in the above example and states a logical equivalence between the result of applying a λ-expression to a term and the result of substituting that term for the variable bound by λ in the formula that the λ-expression is derived from. This rule, which governs the logico-semantic structure of λ-expressions, is usually called *λ-conversion*. The left-to-right part is sometimes referred to as *λ-contraction* or *λ-reduction*, while the right-to-left part is sometimes called *λ-abstraction*.

While the mechanism of λ-conversion is conceptually quite simple, concrete examples can get quite complicated. Consider trying to simplify the following formula.

(6) $\forall x[\lambda z[\lambda w[[Q(w) \wedge B(m)] \leftrightarrow K(x, w)](z)](j) \rightarrow \lambda y[K(y, x) \vee Q(y)](m)]$

To find what the λ-reduced form of (6) is, it is crucial to parse (6) correctly. One must find the scope of each occurrence of the λ-operator and identify the individual term it is predicated of. A tree diagram like the one in (7) might help.

(7)

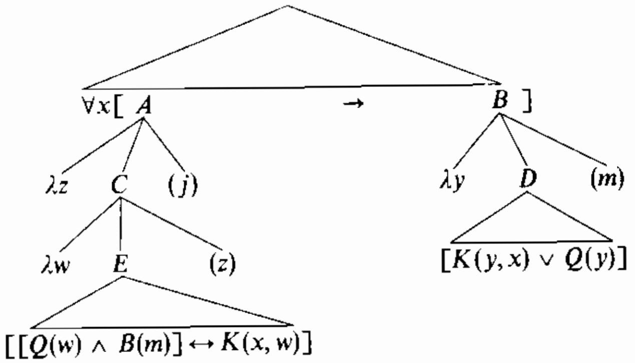
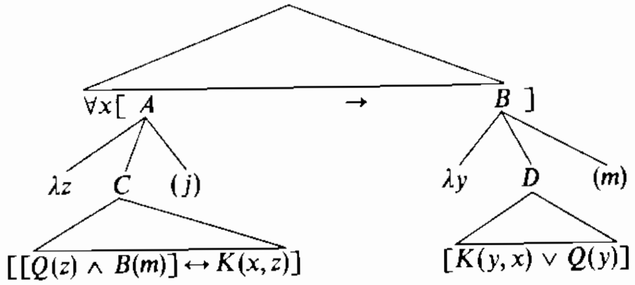


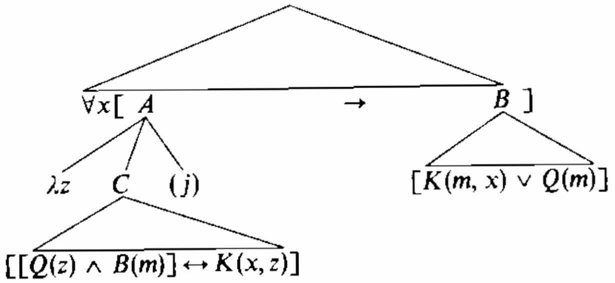
Diagram (7) exhibits the syntactic structure of (6). The topmost node represents the entire formula, and each capital letter represents one of its subformulas. Whenever we find a subtree of the form $[\lambda\alpha\delta(t)]$, we can uniformly substitute t for α in δ. If we start doing this bottom up in (7), after the first step we get the following:

(8)



We reduced the subtree rooted in C by substituting z for w in E. If we do the same for the subtree rooted in B, we obtain the following:

(9)



Finally, we repeat the same operation on the subtree rooted in A, and we obtain the correct reduced form of (6), as follows:

(10) $\forall x[[[Q(j) \wedge B(m)] \leftrightarrow K(x, j)] \rightarrow [K(m, x) \vee Q(m)]]$

Exercise 1 Reduce the following formulas as much as you can. Show each step in the derivation.

(*a*) $\lambda x[\exists z[\lambda y[K(x, y)](z) \wedge R(z, x)]](j)$

(*b*) $\lambda y[\lambda x[K(x, y)](j)](m)$

(*c*) $\lambda z[\lambda x[[K(x, z) \wedge R(x, z)] \vee R(z, x)](j)](m)$

(*d*) $\exists y[\lambda z[\lambda x[B(x) \rightarrow \exists w[R(x, w)]](j) \wedge \lambda x[B(x) \vee Q(x)](z)](y)]$

So far we have considered the syntax of the λ-operator and stated λ-conversion as a syntactic principle. We must now provide a semantics for λ-terms that makes the principle in (5) valid. This is done in (11). For simplicity we will omit here and throughout this chapter explicit reference to the context coordinate c.

(11) $[\![\lambda x\psi]\!]^{M,w,i,g} = \{u \in U : [\![\psi]\!]^{M,w,i,g[u/x]} = 1\}$

In (11) we specify the extension of $\lambda x\psi$ in a model M, in a world w, at an instant i, and with respect to assignment g. Since we are dealing with a one-place predicate, such an extension will be a set. Which set is determined by successively computing the value of ψ for every u as a value of x. All those individuals that make ψ true at $\langle w, i\rangle$ will be in the set.

As an example, consider M_4 and g_4 on pages 216–217. Let us compute $[\![\lambda x[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4}$. This is going to be equal to $\{u \in U_4 : [\![[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4[u/x]} = 1\}$. As there are three individuals in U_4, we have to compute $[\![[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4[u/x]}$ three times, interpreting x as a, as b, and as c. By performing the relevant computations, we get $[\![[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4[a/x]} = 0$, $[\![[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4[b/x]} = 1$, and $[\![[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4[c/x]} = 1$. Therefore, $[\![\lambda x[P(x) \wedge \neg x = m]]\!]^{M_4, w', i', g_4} = \{b, c\}$.

Exercise 2 (1) Give the extensions of $\lambda x[P(x) \wedge \neg x = m]$ at every world and time of M_4.

(2) Give the extension of the following formulas in M_4 with respect to g_4 in the indicated worlds and times.

(*a*) $\lambda x[P(x)]$ w', i'''

(*b*) $\lambda x[x = y]$ $w'', i'; w'', i''; w', i'$

(*c*) $\lambda x[P(j)]$ $w', i''; w'', i'; w'', i'''$

(*d*) $\lambda x[P(x) \rightarrow [x = m \vee x = j]]$ $w', i'; w', i'''$

It turns out that the semanties in (11) validates λ-conversion but only in a restricted form. More specifically, there are two types of cases in which the result of applying a λ-expression to a term t will not be equivalent to its reduced counterpart. The first type of case is when a variable clash arises as a result of the substitution. To see what this amounts to, let us consider a specific example. Take a simple model M_8 for IPC such that $M_8 = \langle U_8, W_8, I_8, <_8, V_8\rangle$, where $U_8 = \{a, b, c\}$, $W_8 = \{w\}$, $I_8 = \{i, i'\}$, $<_8 = \{\langle i, i'\rangle\}$, and $V_8(K)(\langle w, i\rangle) = V_8(K)(\langle w, i'\rangle) = \{\langle a, b\rangle, \langle b, c\rangle\}$. Let us furthermore adopt an assignment function g' such that $g'(y) = b$. Under these assumptions the following facts obtain:

(12) *a.* $[\![y]\!]^{M_8, w, i, g'} = b$

b. $[\![\lambda x[\exists y[K(y, x)]]]\!]^{M_8, w, i, g'} = \{b, c\}$

The property of being an x such that something stands in the relation K with x has the set $\{b, c\}$ as its extension in M_8 at $\langle w, i\rangle$. Moreover, under assignment g', the variable y denotes b. Let us now apply $\lambda x[\exists y[K(y, x)]]$ to y:

(13) *a.* $\lambda x[\exists y[K(y, x)]](y)$

b. $\exists y[K(y, y)]$

Intuitively, (13*a*) should be true, since something (namely *a*) stands in the K relation to b, which is the individual that y happens to denote. However the contracted form of (13*a*) is (13*b*), and (13*b*) says nothing at all about b; it says that something stands in the K relation with itself, which in M_8 at $\langle w, i\rangle$ is false. It is pretty clear what goes wrong in going from (13*a*) to (13*b*): the final y is a free variable in (13*a*), but in the derivation of (13*b*), that occurrence of y gets accidentally bound by the existential quantifier. We can't allow this to happen. This means that the principle of λ-conversion must be qualified accordingly:

(14) $\lambda x[\psi](t) \leftrightarrow \psi[t/x]$, where t is any term and $\psi[t/x]$ is the result of substituting t for all occurrences of x in ψ, *unless t* is a variable that becomes bound as a result of the substitution.

There is an easy way of getting around the problem. As we know from chapter 3, our semantics allows alphabetie changes of bound variables in formulas. That is, if we have a formula ψ that contains n bound occurrences of a variable x and z is a variable not occuring in ψ, the formula that results from substituting z for x in ψ will be equivalent to ψ. Thus, for example, (13*a*) is equivalent to (15*a*).

(15) *a.* $\lambda x[\exists z[K(z, x)]](y)$

b. $\exists z[K(z, y)]$

And (15*a*) then reduces unproblematically to (15*b*). Thus whenever the risk of a variable clash arises we can always avoid it by judicious use of alphabetic variants.

The second class of restrictions on λ-conversion arises in connection with model and intensional contexts. Consider, for example, (16).

(16) *a.* $\lambda x[\Box P(x)](m)$

b. $\Box P(m)$

Suppose that m is a rigid designator. Then (16*b*) says that in every possible world, m has property P. Yet (16*a*) says that the individual who happens to be m in the world of evaluation has property P in every world. Since m always picks out the same individual at all worlds, this will amount to saying that that individual has property P in every world. So (16*a*) and (16*b*) are logically equivalent if m is a rigid designator.

Suppose, on the other hand, that m is not a rigid designator, but something like, say, "Mr. Muscle." Then (16*b*) says that in every world Mr. Muscle, whoever he may be in that world, has property P. But (16*a*) says something quite different. It says that the individual who in fact *happens* to be Mr. Muscle (in the world of evaluation) has property P in every world. Conceivably, (16*a*) might be true, and (16*b*) false, or vice versa. Hence, if m is not a rigid designator, (16*a*) and (16*b*) are not logically equivalent.

Exercise 3 Verify the latter claim by (*a*) evaluating (16*a*) and (16*b*) in M_4, at w', i'' and (*b*) constructing a model where (16*a*) is true and (16*b*) false.

Thus if a formula ψ contains intensional contexts, (14) will be valid only if the term t is a rigid designator. In other words we have the following:

(17) $\exists y[\Box y = t] \rightarrow [\lambda x[\psi](t) \leftrightarrow \psi[t/x]]$

where ψ contains intensional operators and $\psi[t/x]$ is as in (14).

The above considerations are a further illustration of how model theory can help in bringing logical syntax into sharper focus. We now turn to a discussion of some possible linguistic uses of the λ-operator.

2 Semantics via Translation

In chapter 3, section 3, we discussed the role of semantic representations in a theory of meaning. The conclusion we tentatively arrived at was that while semantics must relate natural language expressions to extralinguistic entities (as on the truth-conditional approach), there might be some advantage in doing so by an intermediate step that links natural language to a logic. In such a view the semantic component of a grammar can be thought of as a map from English onto a logic whose truth-conditional embedding in the world is known. The map onto logic provides us with a convenient representational medium that can perhaps facilitate the specification of semantic generalizations. And at the same time it provides us with an indirect characterization of truth conditions.

However, it was impossible to come up with a simple compositional map onto IPC for the fragments we have considered. To develop such a map, one needs, for example, a way of translating into IPC VPs like *loves Pavarotti* in terms of the translations of *love* and *Pavarotti*. But *love Pavarotti* is a complex (or derived) property, and we had no systematic way of defining properties. The addition of the λ-operator to IPC enables us to do so and thus provides the base for formulating a compositional interpretive procedure that takes the form of a systematic algorithm translating from the relevant level of syntactic representation (in the case at hand, LF) into a suitable logical language (in the case at hand, IPC).

In what follows, we will first show how this can be done for the fragment of English developed so far and then discuss some general consequences of this way of proceeding. (To keep things simple, we ignore the contextual and discourse considerations discussed in chapter 6.) We repeat here the relevant rules of F_3, for ease of reference.[1]

(18) *a.* S → NP Pred
b. S → S conj S
c. Pred → INFL VP
d. VP → V_t NP
e. VP → V_i
f. VP → V_dt NP PP[to]
g. INFL → (NEG) $\left\{\begin{matrix}\text{PAST}\\ \text{PRES}\\ \text{FUT}\end{matrix}\right\}$ 3rd SNG
h. NP → Det Nom
i. PP[to] → to NP
j. Det → the, a, every
k. N → Pavarotti, Loren, Bond, ..., $x_{n[\text{PRO},\text{FEM},\dots]}$, ...
l. Nom → book, fish, man, woman, ...
m. V_i → be boring, be hungry, walk, talk, ...
n. V_t → like, hate, kiss, ...
o. V_dt → give, show, ...

p. conj → and, or
q. NP → N
r. $\bar{\text{S}}$ → COMP S
s. VP → $\text{V}_\text{S}\ \bar{\text{S}}$
t. V_S → believe, know, regret, ...
u. COMP → that

The rules for quantifier raising and INFL raising are given in (19) and (20).

(19) $[_\text{S}\ X\ \text{NP}\ Y] \Rightarrow [_\text{S}\ \text{NP}_\text{i}\ [_\text{S}\ X\ e_i\ Y]]$

Here NP = [Det Nom] and X and Y are the rest of the sentence.

(20) $[_\text{S}\ \text{NP INFL}\ X] \Rightarrow [_\text{S}\ \text{INFL}[_\text{S}\ \text{NP}\ X]]$

The translation map onto IPC is defined recursively. First we state a correspondence between the categories of F_3 and those of IPC. Intuitively, this correspondence determines the logical type of the various expressions (it determines what kind of semantic entity will ultimately correspond to expressions of various syntactic categories).

(21)

F_3	IPC
N	individual terms (variables and constants)
V_i (intransitive verbs)	Pred_1 (one-place predicates)
Nom (common nouns)	Pred_1
V_t (transitive verbs)	Pred_2 (two-place predicates)
V_dt (ditransitive verbs)	Pred_3 (three-place predicates)
V_S (believe-type verbs)	relations between individual terms and propositional terms

We now have to specify a translation for the lexical entries of F_3. A restricted number of such entries have logical particles as their meaning. These are listed in (22). (We will use α' as an abbreviation for "the translation of α into IPC.")

(22) NEG = $\neg$
and′ = $\wedge$
or′ = $\vee$
FUT′ = **F**
PAST′ = **P**
that′ = $^\wedge$
$t'_n = x_n$, where t_n is a trace or a pronoun

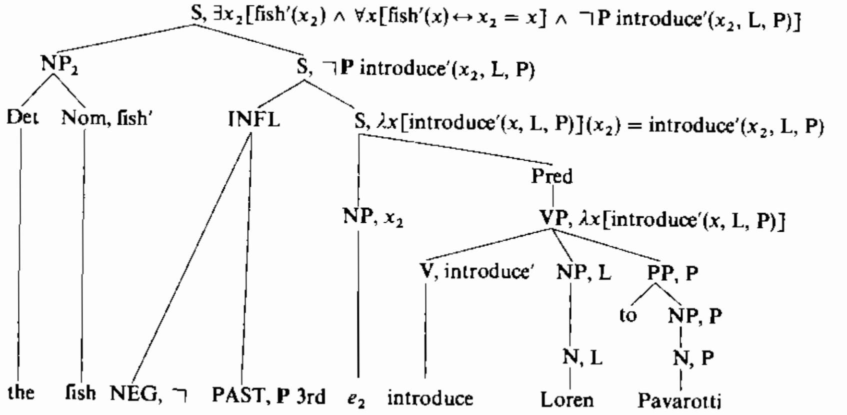
Notice that for convenience we include traces and pronouns among the "logical" components of the terminal vocabulary of F_3. For nonlogical lexical entries we adopt the following convention:

(23) If α is of lexical category A, α' is a constant of IPC of the appropriate type as defined by (21).

We come next to the recursive part of the translation map. We design the map so that it assigns a translation to each node of an LF tree in terms of the translation of its daughters. Since the leaves of a well-formed LF tree are lexical entries and we have just assigned a translation to lexical entries, we are guaranteed that the translation map will eventually arrive at a determinate value. Throughout, for any category A, A' denotes the translation of the (sub)tree rooted in A.

(24) *a.* If $\Delta = [_A\ B]$ or $\Delta = [_A \text{ to } B]$, then $\Delta' = B'$
b. If $\Delta = [\text{NP Pred}]$, $\Delta' = \text{Pred}'\,(\text{NP}')$
c. If $\Delta = [\text{S}_1 \text{ conj } \text{S}_2]$, $\Delta' = \text{S}_1' \text{ conj}' \text{ S}_2'$
d. If $\Delta = [\text{V NP}]$, $\Delta' = \lambda x[\text{V}'(x, \text{NP}')]$
e. If $\Delta = [\text{V NP PP}]$, $\Delta' = \lambda x[\text{V}'(x, \text{NP}', \text{PP}')]$
f. If $\Delta = [_{\bar{\text{S}}} \text{ COMP S}]$, then $\Delta' = \text{COMP}'\ \text{S}'$
g. If $\Delta = [\text{NP}_i \text{ S}]$, then
if $\text{NP}_i = [\text{every } \psi]_i$, $\Delta' = \forall x_i\ [\beta'(x_i) \rightarrow \text{S}']$
if $\text{NP}_i = [\text{a } \psi]_i$, then $\Delta' = \exists x_i[\beta'(x_i) \wedge \text{S}']$
if $\text{NP}_i = [\text{the } \psi]_i$, then $\Delta' = \exists x_i[\beta'(x_i) \wedge [\forall y[\beta'(y) \rightarrow x_i = y] \wedge \text{S}']]$
h. If $\Delta = [\text{INFL S}]$ and TNS = PRES, PAST, or FUT, then
if INFL = PRES AGR, then $\Delta' = \text{S}'$
if INFL = PAST AGR, then $\Delta' = \text{PAST}'\ \text{S}'$
if INFL = FUT AGR, then $\Delta' = \text{FUT}'\ \text{S}'$
if INFL = NEG TNS AGR, then $\Delta' = \text{NEG}'\ [\text{TNS AGR S}]'$

As an example, consider "The fish did not introduce Loren to Pavarotti." (We abbreviate "Loren′" as "L" and "Pavarotti′" as "P" respectively.)


S, ∃x₂[fish′(x₂) ∧ ∀x[fish′(x) ↔ x₂ = x] ∧ ¬P introduce′(x₂, L, P)]
NP₂
S, ¬P introduce′(x₂, L, P)
Det
Nom, fish′
INFL
S, λx[introduce′(x, L, P)](x₂) = introduce′(x₂, L, P)
Pred
NP, x₂
VP, λx[introduce′(x, L, P)]
V, introduce′
NP, L
PP, P
to
NP, P
N, L
N, P
the
fish
NEG, ¬
PAST, P 3rd
e₂
introduce
Loren
Pavarotti

In this example we find an IPC translation as defined by our algorithm next to each node of the above LF tree. Not every node will have a translation. For example INFL does not, as it contains diverse elements, like negation and tense. These elements could be amalgamated into a unit by means of the λ-operator, but we will not try to do that here. NP nodes of the form Det Nom also lack a translation (although, of course, their contribution to meaning is represented). This reflects the fact that IPC lacks a category that corresponds to such NPs. Similarly, the Det nodes inside these NPs lack an IPC translation because of the absence of an appropriate IPC category. These omissions reflect the fact that we don't know at this point what sorts of semantic entities NPs and Dets correspond to, and this prevents us from providing a fully compositional way of interpreting NPs (see chapter 3 for relevant discussion and chapter 9 for a possible solution). Sometimes the translation associated with a node can be simplified, in which case both the unreduced and reduced forms are given.

Exercise 4 Give the translations of the various LF structures associated with "Every fish will not like a man."

On an approach along the preceding lines, semantics is split into two components. First, a compositional translation into a logical calculus is provided. Second, this logical calculus is equipped with a truth-conditional and model-theoretic semantics. As we have seen, it is possible to provide truth conditions directly for LFs, and in this sense, translation into a logical calculus appears to be dispensable. What such a translation provides us with is a very explicit way of representing the truth conditions associated with English sentences. Of course, the calculus must be rich enough to support such an enterprise. The λ-operator is quite useful in this regard, as it enables us to represent compositionally complex VPs in our logical syntax. In IPC without λ-abstraction we had no simple way of doing that.[2]

In what follows, we will reserve the term *logical form* (with lowercase *l* and lowercase *f*, which we shall abbreviate as *lf*) for the calculus we use to translate LF into. "The logical form of a sentence S, relative to its LF structure *A*" is taken to mean "the translation of *A* into the semantic calculus *lf*."

This kind of two-stage semantics calls for a few general remarks. So far even though we have occasionally found it convenient to translate English into a logical calculus, we had no general procedure for doing so. The "correct" translation had to be decided on a case by case basis. In such a situation there is no way of knowing whether the translation procedure is finitely specifiable, whether it represents a viable tool for characterizing our semantic competence. Now we see that a translation procedure for a fragment of English can be systematically and compositionally specified. We are thus entitled to hope that this method can be extended to larger fragments, that the relation between syntactic structure and logical structure in general is rule-governed.

If one views logical form as a component of grammar, some of the central questions about meaning can be reformulated as follows: what are the general principles that govern the relation between the relevant level of syntax (LF) and logical structure (*lf*)? Is there a finite repertoire of translation schemata that grammar uses? Do these translation schemata account for properties of language that various levels of syntactic structure leave unaccounted for? Much current semantic research is centered around these issues, but we cannot address them here.

Is it possible to regard logical form so construed as providing us with a theory of semantic representation, with a theory that characterizes what we grasp in processing a sentence? This question is very controversial. For many the answer is no.[3] We think it is possible, on the basis of the following considerations. What are some of the standard requirements that a theory of semantic representation is expected to fulfill? Minimally, grasping a sentence must involve recovering some representation of its lexical components and some way of amalgamating representations of lexical meanings into representations of the meaning of larger units, until a representation of the meaning of the whole sentence is recovered. Whichever format one eventually chooses for representing lexical meaning, a theory of semantic representation must support this compositional process. Our semantics characterizes precisely that: the compositional process that lexical meanings have to support if they are to do their job of systematically contributing to the meaning of larger units. Moreover, any semantic representations must account for the semantic intuitions of speakers. Given the representations associated with two sentences *A* and *B*, we must be able to tell whether they are contradictory, compatible, or equivalent in terms of their representations at some level. If one uses a logical calculus in the way we are here, equivalence (content synonymy), contradiction, and other semantic relations can be syntactically checked using axioms and rules of inference associated with the calculus.

Furthermore, the particular approach that we are developing, even if it is only a rough first approximation, does embody specific empirical claims about the nature of semantic representation. They can be summarized as follows.

- It has a Boolean core, which simply means the following: one can isolate certain logical words (*and*, *or*, *not*) with the logical properties we have discussed. These elements constitute the central devices for gluing together stretches of information. They are related according to the "laws of thought" originally characterized as such by the mathematician George Boole (on which classical logic is based). Much recent work has been devoted to showing that this Boolean core determines the logical structure of every syntactic category (and not just S).[4]
- It incorporates a categorization of semantic objects: individuals, properties, and propositions that are related to each other in a certain way and have a certain structure (the structure of propositions is Boolean).
- It has devices to express quantification. Some forms of quantification make use of a variable-binding mechanism; others (like the modals) do not.
- It includes indexical expressions and other context-sensitive features.

The apparatus developed so far is a way of characterizing these properties while meeting the general criteria that any theory of meaning should satisfy. Even though many problems are left open, we think that any adequate theory of semantic representation should incorporate the features we have listed.

If it is justified to regard the present theory as a way of characterizing *what* one grasps in processing a sentence, it does not seem to be equally justified to view the present approach as a characterization of *how* meaning is grasped.[5] The present theory does not make any claim as to the specific psychological processes that form the actual mechanisms of comprehension. The relation between semantics and meaning comprehension is just an instance of the relation between grammar, viewed as an abstract rule system, and language processing. Such a relation is not straightforward, although ultimately one would like to see these theories converge to an integrated theory of cognition. What we are trying to do here is to develop a theory of the structural properties that a speaker's representations of meaning must have.

While a logical calculus with the above characteristics might well be the basis of the way meaning is mentally represented, we also believe that this is so only because such a calculus is embeddable in the world. Mental representations represent reality: individuals having properties and standing in relations. Our strategy has been to characterize what such representations are representations of. The logical structure of our calculus is lifted from, and supported by, the denotational structure of the model.

As we have pointed out on various occasions, if we didn't have a way of relating our logical syntax to what it represents, we would have simply mapped configurations of symbols (natural language) onto other configurations of symbols (a logical syntax). We would not know why language can convey information about things that are not configurations of symbols.

There is also a further reason why the truth-conditional interpretation cannot be left out of the picture, as pointed out in chapter 3, section 3. What is it that makes a logical syntax logical? As far as we can see, one of two things. Either such a syntax is associated with a semantics in terms of which some relevant relation of content synonymy is defined, or it is associated with inference rules that define some relevant notion of equivalence among syntactic structures. But to do their job, inference rules must be sound, and the most general way to show that they are sound is to provide a denotational semantics such as we have specified for them. Thus in either case a level of denotational semantics seems to be a necessary condition for the success of the enterprise.

3 Relative Clauses

Consider restrictive relative clauses such as the following italicized phrases:

(25) *a.* a student *whom Mary thinks she likes* ____
b. the boy *that John believes* ____ *came late*
c. the woman *who* ____ *lives next door*

If one were to characterize relative clauses in theory-neutral terms, one could say that they are predicates derived from sentences. Typically, such derived predicates are used to modify a head noun (*student* and *boy* in (25*a*, *b*)). Strategies for relative clause formation often involve the use of gaps or resumptive pronouns. English uses the former strategy. In (25*a*–*c*) a dash indicates the position of the gap. English also employs relative pronominals (such as *who*) dislocated to sentence-initial position.

The idea of deriving predicates from sentences essentially characterizes the λ-operator. This strongly suggests that such an operator may be involved in the interpretation of relative clauses.

To see this, we will add a rudimentary form of relativization to F_3. Within the general syntactic framework that we are using, relative clauses are transformationally derived. For example, (25*a*) is derived from an underlying D-structure like the one in (26) by fronting the *wh* pronoun:

(26) a student [Mary thinks she likes whom]

In the T model of grammar described in chapter 3, section 2.1, *wh* movement is part of the map from deep structure to surface structure. This provides a way of capturing the fact that some properties of the dislocated pronoun are inherited from its DS position (for example, the case of the *wh* element sometimes depends on its DS position; compare in examples (25*a*) and (25*c*) the distribution of *who* versus *whom*). We further assume that various principles (like some version of the coordinate structure constraint, the principle of subjacency, etc.) prevent *wh* movement from overgenerating. Unfortunately, we cannot properly discuss those principles here.[6]

For ease of implementation we will simply assume that the category N contains relative pronouns of the form *that*$_n$, for any number n. Whether *that* should in fact be regarded as a relative pronoun (on a par with *who*) is questionable. Our choice is dictated by the desire to avoid complications stemming from gender and case agreement that would arise if we were to discuss the syntax (and semantics) of *who*. We then add the following syntactic rules to F_3.

(27) *a.* NP → Det Nom*
b. Nom* → Nom S
c. $[_S\ X\ \text{that}_n\ Y] \Rightarrow [_S\ \text{that}_n [_S\ X\ e_n\ Y]]$

X or Y can be empty in (27*c*). Rules (27*a*, *b*) are two new PS rules, and (27*c*), the rule for *wh* movement, is a transformational map that relates D-structures to S-structures. Notice that the addition of (27) to F_3 will allow it to generate ill-formed phrases like *the boy that Mary likes John* (where no gap occurs in the relative clause). There are various ways of dealing with this. Here we will assume that these phrases are uninterpretable in terms of the semantics to be given shortly. We will set up things in such a way that relative clauses will receive a translation (and be interpretable) only if an *indexed* occurrence of the relative pronoun *that*$_n$ is adjoined to S. S in a Nom* constituent will be uninterpretable if there is no sentence-initial relative pronoun.

We add the following translation rules to the semantics for F_3.

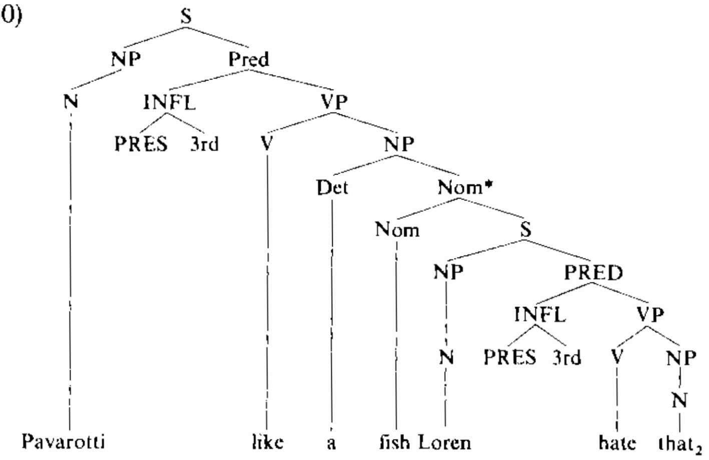
(28) *a.* If $\Delta = [_S\ \text{that}_n\ \text{S}]$, then $\Delta' = \lambda x_n[\text{S}']$.
b. If $\Delta = [\text{Nom S}]$, then $\Delta' = \lambda x[\text{Nom}'(x) \wedge \text{S}'(x)]$.

As an example, consider the following:

(29) Pavarotti likes a fish that Loren hates.

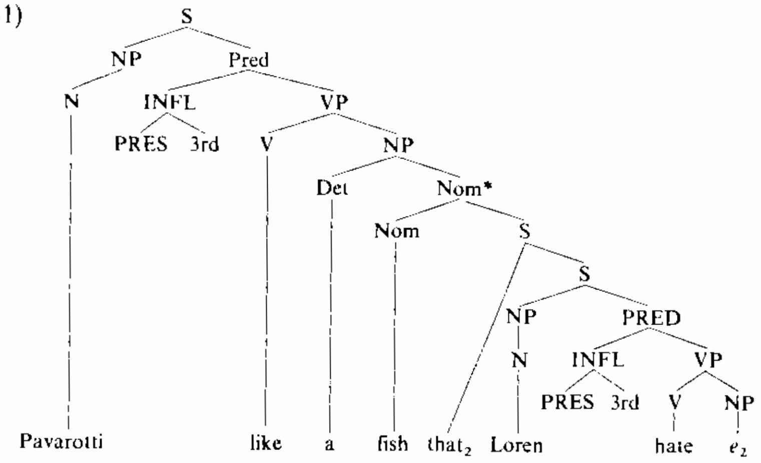
The deep structure of (29) is given in (30).

(30)



The surface structure is as follows:

(31)

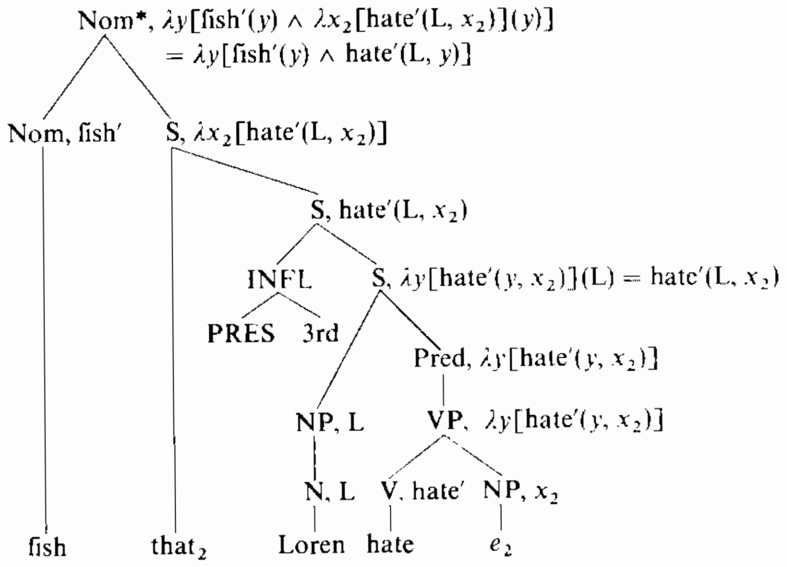


The LF structure (in schematic form) is (32).

(32) $[_S\,[_{NP}$ a fish $[_S$ that$_2$ [Loren hate e_2]]] $[_S$ Pavarotti likes e_4]]

Let us display in full the structure of the relative clause along with its step by step translation.

(33)


Nom*, $\lambda y[\text{fish}'(y) \wedge \lambda x_2[\text{hate}'(L, x_2)](y)]$
$= \lambda y[\text{fish}'(y) \wedge \text{hate}'(L, y)]$
Nom, fish'
S, $\lambda x_2[\text{hate}'(L, x_2)]$
S, $\text{hate}'(L, x_2)$
INFL
S, $\lambda y[\text{hate}'(y, x_2)](L) = \text{hate}'(L, x_2)$
PRES 3rd
Pred, $\lambda y[\text{hate}'(y, x_2)]$
NP, L
VP, $\lambda y[\text{hate}'(y, x_2)]$
N, L
V, hate'
NP, x_2
fish
that$_2$
Loren
hate
e_2

With this interpretation for the nominal *fish that Loren hates*, the translation associated with (32) will follow the usual pattern, which will give us (34). (Again, L = Loren, and P = Pavarotti.)

(34) $\exists x_4[\lambda y[\text{fish}'(y) \wedge \text{hate}'(L, y)](x_4) \wedge \text{like}'(P, x_4)]$
$= \exists x_4[[\text{fish}'(x_4) \wedge \text{hate}'(L, x_4)] \wedge \text{like}'(P, x_4)]$

This seems to represent correctly the truth conditions that (29) intuitively has.

Let us reiterate that the syntactic treatment of relative clauses adopted here is a gross oversimplification. The syntax of relative clauses has been studied quite extensively in a variety of frameworks and languages since the inception of generative grammar and of the various more adequate alternative syntactic approaches (transformational and nontransformational) that succeeded classical transformational grammar. The point we wish to emphasize here is that on any viable approach, something like λ-abstraction will likely be needed for a compositional semantics of relativization. In the framework that we are using, this is particularly evident. The gap left behind by a fronted relative pronoun is interpreted in the usual way as a variable, and the fronted relative pronoun coindexed with the gap seems to act precisely like a λ-abstractor over that variable. Thus, dislocated *wh*-pronouns in English appear to be a very direct syntactic manifestation of such an operator.

4 VP Disjunction and Conjunction

Consider sentences like the following:

(35) *a.* Pavarotti is boring and hates Bond.
b. Pavarotti is hungry or is tired.

In (35*a*, *b*) we have what look like conjoined and disjoined VPs. In the present section we are going to discuss briefly their syntax and semantics.

4.1 Generalizing the scope of logical operators

In the early times of transformational grammar it was proposed that sentences like those in (35) be derived from underlying structures that looked like those in (36) via a transformation called conjunction (or disjunction) reduction.

(36) *a.* Pavarotti is boring, and Pavarotti hates Bond.
b. Pavarotti is hungry, or Pavarotti is tired.

It is hard to come across a precise definition of conjunction reduction in the literature, although the intuitive idea behind it is fairly clear.[7] In a coordinated structure of the form $[\mathrm{S}_1$ and $\mathrm{S}_2]$ parts of S_2 could be deleted if they were identical in certain ways with parallel parts of S_1. Part of the motivation behind such a proposal was the evident synonymy between (36) and (35). So one could propose that, say, (36*a*) is the S-structure (and LF structure) of (35*a*) and that the deletion of the second occurrence of *Pavarotti* takes place in the phonology and therefore doesn't affect meaning.

This simplistic proposal, however, cannot work. Consider the following:

(37) *a.* A man is boring and hates Bond.
b. Every man is boring or is hungry.

(38) *a.* A man is boring, and a man hates Bond.
b. Every man is boring, or every man is hungry.

Clearly (37*a*, *b*) do not have the same meaning as (38*a*, *b*), their alleged sources. More specifically, (37*a*) entails (38*a*), but not vice versa, and (38*b*) entails (37*b*), but not vice versa. If conjunction reduction were a phonological deletion phenomenon that does not affect meaning, one would not expect this pattern to arise.

These considerations have led various researchers to adopt a syntax for VP conjunction and disjunction of the following kind:[8]

(39) VP → VP conj VP

The point then becomes how to interpret conjoined (or disjoined) VPs in such a way that (35*a*, *b*) come out as being equivalent to (36*a*, *b*) but (37*a*, *b*) do not come out as equivalent to (38*a*, *b*), respectively. The λ-operator makes this very easy. For any two predicates P_1 and P_2 of IPC we can define a new operator that, when applied to P_1 and P_2, gives us their conjunction or disjunction. One way of doing this is as follows:

(40) *a.* $[P_1 \wedge P_2] = \lambda x[P_1(x) \wedge P_2(x)]$
b. $[P_1 \vee P_2] = \lambda x[P_1(x) \vee P_2(x)]$

This kind of definition is usually called a pointwise definition. What it does is extend an operation, say $\vee$ (previously undefined as a predicate operator), to any predicates P_1 and P_2 in terms of an already defined sentential operator $\vee$ by looking at the value of $P_1(x) \vee P_2(x)$. We look at the values that $P_1(x) \vee P_2(x)$ gets when we assign to x successively each individual in U. (The domain in U can be regarded as an abstract space in which the individuals contained in U constitute the points, whence the term *pointwise*: we look at the values of $P_1(x) \vee P_2(x)$ point by point, or individual by individual.) The λ-operator makes the job of providing such pointwise definitions extremely straightforward. In this way one can see that the semantic values of predicates (properties) inherit the Boolean structure of propositions. This means, for example, that just as $\neg[\psi \wedge \phi]$ is equivalent to $[\neg\psi \vee \neg\phi]$, the predicate $\lambda x \neg[\psi \wedge \phi]$ will be the same predicate as $\lambda x[\neg\psi \vee \neg\phi]$.

The semantics for VP conjunction and disjunction should by now be obvious:

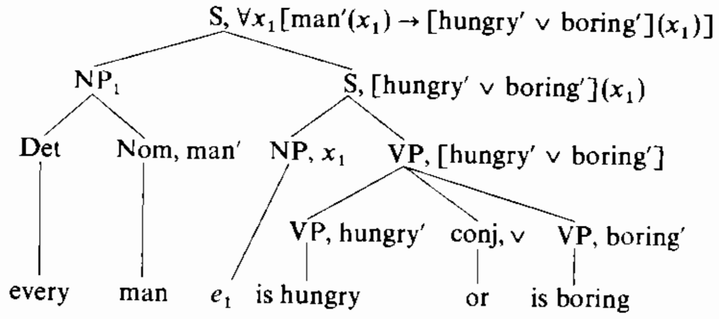
(41) If $\Delta = [\mathrm{VP}_1 \text{ conj } \mathrm{VP}_2]$, then $\Delta' = \mathrm{VP}_1' \text{ conj}' \mathrm{VP}_2'$.

To simplify the discussion of this proposal, let us put aside the syntax of INFL for the time being; that is, let us assume that we are adding rule (39) to the fragment F_2 of chapter 3 (where INFL does not exist) rather than to F_3. Let us further assume that we are interpreting F_2 via translation into IPC along the lines described above, rather than directly. Call the resulting fragment F_2'. Under these assumptions, (42*a*) will have in F_2' the S-structure in (42*b*) and the LF in (42*c*). Each node of the LF structure is associated with its IPC translation.

(42) *a.* Every man is hungry or is boring.

b. $[_S [_{NP}$ every man$][_{VP} [_{VP}$ is hungry$] [_{conj}$ or$][_{VP}$ is boring$]]]$

c.



S, $\forall x_1[\text{man}'(x_1) \rightarrow [\text{hungry}' \vee \text{boring}'](x_1)]$
NP$_1$
S, $[\text{hungry}' \vee \text{boring}'](x_1)$
Det
Nom, man′
NP, x_1
VP, $[\text{hungry}' \vee \text{boring}']$
VP, hungry′
conj, ∨
VP, boring′
every
man
e_1
is hungry
or
is boring

By working through the definition of ∨ in the translation associated with the topmost node in (42*c*), we obtain the following results:

(43) *a.* $\forall x_1[\text{man}'(x_1) \rightarrow \lambda y[\text{hungry}'(y) \vee \text{boring}'(y)](x_1)]$ by (40*b*)

b. $\forall x_1[\text{man}'(x_1) \rightarrow [\text{hungry}'(x_1) \vee \text{boring}'(x_1)]]$ by λ-conversion

This gives us the desired results. It is easy to see that on the present analysis, (42*a*) does not come out as equivalent to "Every man is hungry, or every man is boring."

Exercise 5 Show that (35*a*) and (36*a*) are equivalent in F_2' but (37*a*) and (38*a*) are not.

Thus an analysis that takes advantage of the λ-operator, as along the present lines, enables one to capture in a very simple way certain scope interactions between quantified subjects and VP-level conjunction and at the same time allows for a rather straightforward syntactic approach to the constructions in question.

It has been noted in a number of recent works that conjunction and disjunction are really cross-categorial operators: expressions of virtually any category can be conjoined and disjoined.[9]

(44) *a.* John and every student liked the show. (NP conj NP)

b. Most or every student came. (Det conj Det)

c. John walked in and around the building. (Prep conj Prep)

d. John saw and bought a shirt. (V_t conj V_t)

e. John saw an old and ugly housc. (Adj conj Adj)

One would like to maintain that whatever the syntactic details, one can come up with a uniform meaning for *and* or *or* across all categories. Such a generalized meaning for *and* or *or* might be a good candidate for the linguistic universal stated as follows:

(45) The semantics of all languages will have cross-categorial operators that have the same logical structure as *and* or *or* in English.

Cross-categorial operators of this kind can indeed be defined by generalizing the pointwise definitions of VP conjunction and disjunction given above. But this goes beyond what can be done here.

The simpleminded version of conjunction reduction given above wasn't able to accommodate the scope phenomena in (37) and (38). This suggests that either one should not analyze them using such a transformation (although perhaps such a transformation might still be needed for other phenomena) or a more elaborate version of conjunction reduction should be developed. Even if the latter hypothesis turns out to be workable, it still seems that its semantics will have to be along the same lines as that developed here.

4.2 INFL again

In F_2' we cannot generate sentences like those in (35), since we lack tense and have too rudimentary an approach to negation. However, if we try to join the treatment of VP conjunction of the previous section and the treatment of INFL in F_3, we run into a problem. The problem is that to generate (46*a*) and (46*b*), we have to allow for structures like those in (47).

(46) *a.* John came latc and will leave early.

b. John will come late and will not appreciate the show.

(47) $[_{Pred} [_{Pred}$ INFL VP$]$ and $[_{Pred}$ INFL VP$]]$

But then what would happen to the two INFLs of a conjoined Pred phrase with respect to our rule of INFL raising? For one thing, we know that extraction out of conjoined structures is subject to strong restrictions: it is only possible when an identical constituent is extracted out of the conjuncts across the board, as illustrated in (48).

(48) the boy [[Mary likes who] and [John hates who]] →
the boy who [[Mary likes ——] and [John hates ——]]

This constraint would prevent us from fronting the two distinct INFLs in sentences like (46*a*). Moreover, even if we did permit a double INFL fronting in (46*a*, *b*), the result would not be semantically coherent, for it would mean something like "It was the case that it will be the case that [John come late ∧ John leave early]."

A possible solution would be to provide a way of interpreting INFL *in situ*, that is, in nonraised position. The λ-operator enables us to do so. By using the same lifting technique introduced in the previous seetion, we can define a number of predicate-level operators along the following lines:

(49) For any one-place predicate Q,
a. $[\neg Q] = \lambda x[\neg Q(x)]$
b. $[\mathbf{P}Q] = \lambda x[\mathbf{P}Q(x)]$
c. $[\mathbf{F}Q] = \lambda x[\mathbf{F}Q(x)]$

We can then provide a rule for interpreting unraised INFL along the following lines:

(50) If Δ = [INFL VP] and TNS = PRES, PAST, or FUT, then
if INFL = PRES AGR, Δ' = VP′
if INFL = PAST AGR, Δ' = [PAST′ VP′]
if INFL = FUT AGR, Δ' = [FUT′ VP′]
if INFL = NEG TNS AGR, Δ' = [NEG′ [TNS AGR VP]′]

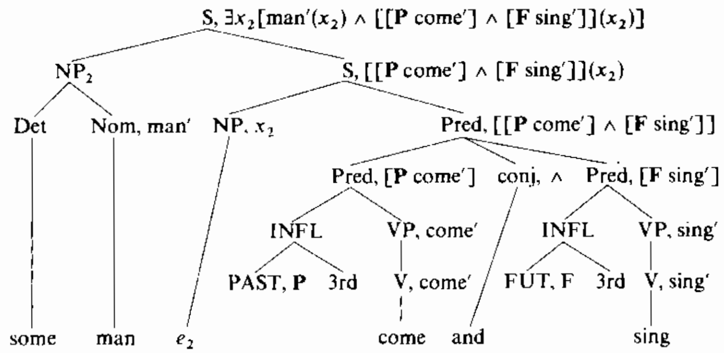
This enables us to interpret sentences like the following:

(51) Some man came and will sing.

The S-structure of (51) is shown in (52*a*). Its LF and semantic interpretation is as in (52*b*).

(52) *a.* $[_{\text{S}}[_{\text{NP}}$ some man$][_{\text{Pred}}[_{\text{Pred}}[_{\text{INFL}}$ PAST 3rd$][_{\text{VP}}$ come$]]$
$[_{\text{conj}}$ and$][_{\text{Pred}}[_{\text{INFL}}$ FUT 3rd$][_{\text{VP}}$ sing$]]]]$

b.



By working through the definitions of the predicate operators in the translation associated with the topmost node in (52*b*), we get the following derivation:

(53) *a.* $\exists x_2[\text{man}'(x_2) \wedge [\lambda y[\mathbf{P}\text{ come}'(y)] \wedge \lambda y[\mathbf{F}\text{ sing}'(y)]](x_2)]$
by (49*b*, *c*)
b. $\exists x_2[\text{man}'(x_2) \wedge \lambda z[\lambda y[\mathbf{P}\text{ come}'(y)](z) \wedge \lambda y[\mathbf{F}\text{ sing}'(y)](z)](x_2)]$
by def. of $\wedge$

c. $\exists x_2[\text{man}'(x_2) \wedge \lambda z[\mathbf{P}\text{ come}'(z) \wedge \mathbf{F}\text{ sing}'(z)](x_2)]$
by λ-conversion
d. $\exists x_2[\text{man}'(x_2) \wedge [\mathbf{P}\text{ come}'(x_2) \wedge \mathbf{F}\text{ sing}'(x_2)]]$ by λ-conversion

This is the desired result. Recall that we are assuming that thc coordinate structure constraint (in whichever way implemented) disallows INFL raising in cases like (51). And notice that we are still maintaining INFL raising as part of the grammar of F_3. We think that this is necessary to obtain the wide scope reading for negation in sentences like (54*a*).

(54) *a.* Every man didn't come.
b. $\forall x[\text{man}'(x) \rightarrow \neg\mathbf{P}\text{ come}'(x)]$
c. $\neg\mathbf{P}\forall x[\text{man}'(x) \rightarrow \text{come}'(x)]$

The present approach predicts that wide scope readings for negation are unavailable in conjoined VPs, as INFL raising in these cases is blocked. This prediction appears to be correct. Sentence (55*a*) has only the reading represented by (55*b*) and not the one represented by (55*c*).

(55) *a.* Every student is tired and isn't enjoying the show.
b. $\forall x[\text{student}'(x) \rightarrow [\text{tired}'(x) \wedge \neg[\text{enjoy the show}]'(x)]$
c. $\neg\forall x[\text{student}'(x) \rightarrow [\text{tired}'(x) \wedge [\text{enjoy the show}]'(x)]$

The topics that we have just addressed are very complex, and our discussion is too brief to really do justice to them. However, it does illustrate, we think, the usefulness of the λ-operator and more generally how various semantic options interact in interesting ways with theoretical choices in the relevant level of syntax.

5 VP Anaphora

In the present section we will discuss an anaphoric phenomenon involving VPs, the one illustrated in (56).[10]

(56) *a.* *A*: Does Bill smoke?
B: No. John does ____.
b. John came but Bill didn't ____.
c. John was interested. The others were ____ too.

We will see that this phenomenon (known as VP anaphora, VP ellipsis, or VP deletion in thc literature) is governed by principles crucially involving something like λ-abstraction.

It should bc noted that VP anaphora is sensitive to surface structure. Consider the following paradigm from Hankamer and Sag (1976):

(57) The children asked to be squirted with the hose, and
a. *we did ______.
b. they were ______.
c. we did it.

We cannot continue the sentence as in (57*a*); we have to use (57*b*) or (57*c*). The reason is that *do* requires an active verb in its complement, and the only available antecedent in (57*a*) is in the passive. This contrasts with some other forms of anaphora for interpreting VPs that do not appear to be subject to the same restriction: (57*c*) illustrates this with *do it* anaphora.

Considerations such as these might lead one to assume that VP anaphora is a deletion process governed by syntactic identity: one is allowed to delete a VP_1 if there is in the context a VP_2 syntactically identical with VP_1. This was in fact the line taken in early generative work on this topic (see Ross (1966)). However, it was later discovered that an approach along these lines runs into serious difficulties. Whether one construes VP anaphora as a deletion process or not, the identity conditions with a contextually specified antecedent cannot be given just in terms of *surface* syntax. They have to be stated at a more abstract level.

Various types of evidence can be used to back up the latter claim. Consider, for example, the following:

(58) Bob left and John will ______.

In (58) the antecedent is in the past, while the missing VP cannot be, as the future auxiliary *will* requires a bare infinitive. The logical representation of (58) is something like (59).

(59) $\mathbf{P}\ \mathrm{leave}'(b) \wedge \mathbf{F}\ \omega(j)$

The ω in (59) can be thought of as a placeholder for a predicate (or as a variable ranging over properties). The interpretive procedure associated with VP deletion must ensure that ω in (59) is interpreted as *leave′*. The point of this example is that the identity conditions licensing VP anaphora must hold at a level where the proper antecedent is present, and surface structure does not seem to be the level. Something like LF or *lf* seems to be a better candidate.

The same point can be made by looking at the following example:

(60) *A*: Did John speak to Bill about himself?
B: Yes. And Frank did ______ too.

The question in (60) is ambiguous. It can either be asking whether John spoke to Bill about John or whether John spoke to Bill about Bill. If the former is the case, then the response in (60) must be taken as saying that Frank spoke to Bill about Frank. If the latter is the case, the response must be interpreted as saying that Frank spoke to Bill about Bill. It is impossible, however, to interpret the question in one way and the response in the other.

If VP anaphora is a deletion process governed by surface syntactic identity, there is no clear reason why such a restriction should obtain. If, on the other hand, VP anaphora is governed by something like identity of meaning, then the restrictions on the interpretation of the response are immediately accounted for. The ambiguity of the question can, in fact, be pinned down to an ambiguity of the VP. The two readings that such a VP has can be represented as in (61).

(61) *a.* $\lambda x[\mathrm{speak}'(x, b, x)]$
b. $\lambda x[\mathrm{speak}'(x, b, b)]$

Property (61*a*) is that of speaking to Bill about oneself. Property (61*b*) is that of speaking to Bill about Bill. The theory of reflexives has to allow for both these possibilities. Whichever interpretation we choose, it will get applied to the response in (60) to give us the following:

(62) *a.* $\lambda x[\mathrm{speak}'(x, b, x)](f) = \mathrm{speak}'(f, b, f)$
b. $\lambda x[\mathrm{speak}'(x, b, b)](f) = \mathrm{speak}'(f, b, b)$

This constitutes the desired result.

We can model the mechanics of the process using either of two strategies. The first, close to the one developed in Sag (1976), maintains the deletion hypothesis but modifies what triggers it. This can be stated roughly as in (63).

(63) Delete a VP_1, if in the context there is a VP_2 whose logical interpretation is identical with that of VP_1.

According to (63) what triggers the deletion is essentially identity of meaning among the relevant constituents. For example, according to (63) the S-structure of (58) is as follows:

(64) [Bob [PAST leave] and John [FUT leave]]

The second occurrence of *leave* having the same meaning as the first occurrence, it can be deleted. This identity of meaning can be stated as requiring either that the LF structure associated with VP_1 be the same as the one associated with VP_2 or that the logical translation of VP_1 (its *lf*) be identical with the logical translation of VP_2 (up to alphabetic variance: two *lf*s that are alphabetic variants of one another count as the same).

Whether we have two distinct ways of stating identity conditions on VP meanings (in terms of LFs or in terms of *lf*s) is an open issue but one that is not going to matter at the elementary level at which we will be addressing the phenomenon.

The second strategy (close to the one developed in Williams (1977)) is the following. One can allow syntax to generate empty VP structures like those in (65):

(65) *a.* $[_{\mathrm{VP}}\, \varnothing]$
b. $[_{\mathrm{VP}}\, [_{\mathrm{V}}\, \varnothing]\, [_{\mathrm{NP}}\, \varnothing]]$

Then one can state an interpretive procedure that requires these empty VPs to be interpreted exactly like some VP in the context. Again there are a variety of ways in which such an interpretive procedure can be stated. One can copy the LF of the antecedent into the empty VP structure, or one can assign to the empty structure the same logical translation as that of the antecedent. And various other options are conceivable.

We present these two approaches partly because most of the research on this topic has been centered around them and partly because it is useful to have a specific mechanism in mind in discussing these matters, even though what we say here will not enable us to choose between the two strategies. For this reason we will lump the two strategies together and refer to them jointly as the semantic theory of VP anaphora. The general point that we wish to make is that both the interpretive and the revised deletion strategy essentially require semantic identity between two constituents, as opposed to identity of surface structures. And characterizing the semantic identity of two VP constituents ultimately requires making extensive use of something like λ-abstraction.

Consider the following example:

(66) John $[_{\mathrm{VP}_1}$ thinks that a petition will be sent in from every city], and Mary does _____ too.

Let us focus on the embedded clause of the first conjunct (the complement of *thinks*). There are two scope possibilities for the two quantified NPs contained within such a clause. They give rise to the two meanings in (67) for VP_1.

(67) *a.* $\lambda x[\mathrm{think}'(x, {}^\wedge\exists y[\mathrm{petition}'(y) \wedge \forall z[\mathrm{city}'(z) \rightarrow [\mathrm{from}'(y, z) \wedge$
$\mathbf{F}\ \mathrm{sent\text{-}in}'(y)]]]])]$
b. $\lambda x[\mathrm{think}'(x, {}^\wedge\forall z[\mathrm{city}'(z) \rightarrow \exists y[\mathrm{petition}'(y) \wedge \mathrm{from}'(z, y) \wedge$
$\mathbf{F}\ \mathrm{sent\text{-}in}'(y)]]])]$

The semantic theory of VP anaphora makes a prediction. Whichever of the two possible readings one selects for VP_1, the missing VP must have that same reading. This follows from the assumption that VP anaphora is

licensed by identity of meanings. The deletion approach based on surface syntax predicts instead that it should be possible to interpret VP_1 as in (67*a*) and the missing VP as in (67*b*). But intuitively we see that this is impossible. We therefore have a further piece of evidence in favor of the semantic approach.

One of the consequences of the semantic approach is that it requires the assignment of scope to quantifiers at the VP level. This is because of examples like the following:

(68) John likes everyone, and Mary does _____ too.

This sentence means something like John likes everyone and Mary likes everyone. We assume here that the quantifier *everyone* cannot have scope over the whole conjunct, as is attested by the ungrammaticality of sentences like "Mary likes $\mathrm{everyone}_1$, and John hates him_1" on the reading indicated by the subscripts. So the reading that (68) has must have some other source. That is, the meaning of the antecedent for the null VP in (68) has to be something like (69):

(69) $\lambda x \forall y[\mathrm{like}'(x, y)]$

There are a variety of ways in which (69) can be obtained. For example, we might say that VP is an admissible "landing site" for raised quantifiers, as was proposed in May (1985). In other words, we are free to attach quantified NPs either to S or to VP, so that one of the admissible logical forms of the first conjunct in (68) is (70).

(70) $[_{\mathrm{S}}\, \mathrm{John}\, [_{\mathrm{VP}}\, \mathrm{everyone}_3\, [_{\mathrm{VP}}\, \mathrm{like}\ e_3]]]$

Structures like $[_{\mathrm{VP}}\, \mathrm{everyone}_3\, [_{\mathrm{VP}}\, \mathrm{like}\ e_3]]$ would then be generally interpreted as in (69).

Notice what would happen if we assignėd sentential scope to the quantified object in (68):

(71) *a.* $\forall y[\lambda x[\mathrm{like}'(x, y)](j)]$
b. $\forall y[\lambda x[\mathrm{like}'(x, y)](j)] \wedge \lambda x[\mathrm{like}'(x, y)](m)$

Formula (71a) represents the meaning of the first conjunct in (68), and $\lambda x[\mathrm{like}'(x, y)]$ represents the meaning of the VP in that conjunct on the assumption that the object is assigned sentential scope. According to the semantic theory of VP anaphora, $\lambda x[\mathrm{like}'(x, y)]$ must be the meaning of the missing VP in the second conjunct. The semantic theory thus assigns the logical structure given in (71*b*) as the meaning of the whole sentence in (68). But here we run into a problem. The variable y is bound in the first conjunct and free in the second. Thus it will be interpreted differently in the first and in the second conjunct. Consequently, the meaning of the missing VP will not be the same as the meaning of its antecedent, which goes against the basic idea of the semantic approach to VP anaphora. Thus we must say that in assigning an antecedent to a missing VP, no variable bound in the antecedent can be free in the interpretation of the missing VP. This in turn implies that we will be able to interpret (68) *only if* we assign VP scope to the object in the first conjunct.

The restriction we just discovered has a large number of perhaps unexpected empirical consequences. Let us start with some fairly immediate ones. Consider (72):

(72) *a.* Some student hates every professor.
b. $\exists y[\text{student}'(y) \wedge \lambda x \forall z[\text{professor}'(z) \rightarrow \text{hate}'(x, z)](y)]$
c. $\forall z[\text{professor}'(z) \rightarrow \exists y[\text{student}'(y) \wedge \lambda x[\text{hate}'(x, z)](y)]]$

By a line of reasoning fully parallel to the reasoning above concerning (68), the VP of (72*a*) can license VP anaphora only if the universal quantifier is assigned VP scope, that is, only on the construal in (72*b*). On all other construals, which assign sentential scope to the object NP in (72*a*), we run into the problem just discussed in connection with (68). Consequently, while (72*a*) *is* ambiguous between readings (72*b*) and (72*c*), the following sentence is predicted not to be:

(73) A student hates every professor, and a secretary does ______ also.

We expect (73) to have only the reading where a single student and a single secretary hate every professor. This seems to be right.

Furthermore, we also expect (74*a*) to be ambiguous and (74*b*) not to be, again by the same argument.

(74) John thinks that a student hates every professor,
a. and Mary does ______ too.
b. and Mary thinks that a secretary does ______ too.

This expectation too appears to be borne out by the facts. What warrants it is that both the readings represented below in (75*a*) and (75*b*) are possible antecedents for the missing VP in (74*a*), but only (75*c*) is available for the missing VP in (74*b*), in view of the restriction on bound variables discussed above.

(75) *a.* $\lambda x[\text{think}'(x, {}^\wedge\exists y[\text{student}'(y) \wedge \forall z[\text{professor}'(z) \rightarrow \text{hate}'(y, z)]]]$
b. $\lambda x[\text{think}'(x, {}^\wedge\forall z[\text{professor}'(z) \rightarrow \exists y[\text{student}'(y) \wedge \text{hate}'(y, z)]]]$
c. $\lambda x \forall z[\text{professor}'(z) \rightarrow \text{hate}'(x, z)]$

These facts are quite intricate, and the semantic theory of VP anaphora seems able to provide a simple and insightful explanation for them. Again

notice that such an explanation does not appear to be available if what triggers VP deletion is identity of VPs at surface structure.

There are some interesting counterexamples to the generalization above pointed out in Hirschbühler (1982). They have to do with sentences like the following:

(76) A Canadian flag was hanging in front of every window. An American flag was too.

Here the object of the prepositional phrase can be naturally understood as having scope over the subject in both conjuncts, contrary to what one would expect on the basis of the discussion above. At present these examples are not well understood. See Hirschbühler (1982) for a relevant discussion.

The following considerations from Williams (1977) lend further support to the semantic theory of VP anaphora. Consider tbe following sentence:

(77) John saw everyone before Mary did.

Sentence (77) has two readings. According to the first, for every person x, John saw x before Mary did. According to the second, John saw everyone, and after that Mary did. Let us call this second reading the group reading. The two readings are not equivalent. The first situation could obtain without the second being the case (but not vice versa: if the second obtains, then the first will obtain also). These two readings are precisely those expected on the present approach. The second or group reading is obtained when the quantified NP *every man* is assigned VP scope in the main clause in (77), as illustrated below:

(78) *a.* $\lambda x \forall y[\text{see}'(x, y)](j)$ BEFORE $\omega(m)$
b. $\lambda x \forall y[\text{see}'(x, y)](j)$ BEFORE $\lambda x \forall y[\text{see}'(x, y)](m)$

We assume that *before* is interpreted as a two-place sentence operator ("ϕ BEFORE ψ" is true iff ϕ is true before ψ is true). The variable on the right-hand side of (78*a*) gets interpreted in terms of the VP meaning represented on the left-hand side, which thereby yields (78*b*).

However, it is also possible to assign sentential scope to the quantified NP *everyone*. In this way *everyone* ends up having scope over the adverbial *before* clause (which is presumably adjoined either under the VP or under the S). The resulting reading is represented in (79*a*).

(79) *a.* $\forall x[\lambda y[\text{see}'(y, x)](j)$ BEFORE $\omega(m)]$
b. $\forall x[\lambda y[\text{see}'(y, x)](j)$ BEFORE $\lambda y[\text{see}'(y, x)](m)]$

The only available VP meaning in this case is $\lambda y[\text{see}'(y, x)]$ in (79*a*). Nothing prevents us from assigning it to ω, for the variable y will still be in the scope of the universal quantifier, as (79*b*) shows. We are *not* using a VP meaning that contains a bound variable and turning it into a free one in interpreting the missing VP. So (79*b*) should be a possible interpretation for (77):

But now imagine continuing (77), repeated in (80*a*), as in (80*b*).

(80) *a.* John saw everyone before Mary did.
b. Yes, and Bill did before Sue did.

It seems that both missing VPs in (80*b*) can only have the group reading. This follows from the analysis above. The easiest way to see this is in terms of the interpretive strategy outlined above (although the same result would also follow from the strategy of deletion under identity of meaning). Notice first that the discourse in (80), even if it involves different speakers, quite clearly forms a coordinate structure, as is attested by the presence of *and* in (80*b*). This means that the quantifier *everyone* in (80*a*) cannot have scope over (80*b*). The logical form of (80*b*) can be represented as in (81).

(81) $\omega(b)$ BEFORE $\omega(s)$

The question is, What are the antecedents available for the predicate variables in (81)? Only two: $\lambda x\forall y[\text{see}'(x, y)]$ in (78*a*) and $\lambda y[\text{see}'(x, y)]$ in (79*a*). If we choose the VP meaning in (79*a*), there is no problem, for we obtain (82).

(82) $\lambda x\forall y[\text{see}'(x, y)](b)$ BEFORE $\lambda x\forall y[\text{see}'(x, y)](s)$

This gives us the group reading. If, however, we choose the VP meaning in (79*a*) as first clause, we get (83).

(83) $\lambda y[\text{see}'(y, x)](b)$ BEFORE $\lambda y[\text{see}'(y, x)](s)$

But in doing so, *x*, a variable bound in the first clause, turns out to be free in the interpretation of the missing VPs, which we know on independent grounds to be impossible. And there is no way to rescue this interpretation on the assumption that quantifiers cannot have scope over a coordinated structure. Thus, (80*b*) is unambiguous.

To sum up, we have considered some of the basic properties of VP anaphora and argued that whether it is viewed as a deletion or as a purely interpretive phenomenon, it requires identity of VP meanings. We have described a simple approach along this semantic line that has quite a number of interesting empirical consequences. The identity conditions on VP meanings can be spelled out at either of two levels. We can spell out VP meanings at the LF level (under a suitable direct model-theoretic interpretation of LF) by requiring that the two VPs be assigned the same LF structure. Or we can spell them out in terms of *lf* by requiring that VPs be assigned the same logical translations (again with ultimately the same model-theoretic values). It is hard to see how the kind of semantics for VPs required by VP anaphora could be given without resorting to λ-abstraction or to a device substantially equivalent to it.

Exercise 6 Add to F_3 the following rules for adjectives:

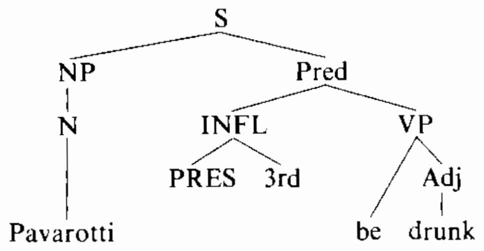
(1) VP → be Adj
Adj → red, drunk, dead, round, blond, . . .

Let us now give a semantics for the newly introduced constructions by means of translation in the style of section 2.

(2) Members of the syntactic category Adj are translated as constants of category Pred_1 of IPC.
If $\Delta = [\text{be Adj}]$, then $\Delta' = \text{Adj}'$

Together (1) and (2) generate sentences like (3*a*) and assign to them the reduced translation given in (3*c*).

(3) *a.* Pavarotti is drunk.
b.



c. drunk′(P)

(*a*) Add the syntactic and semantic (translation) rules for prenominal adjectives. Use λ-abstraction in introducing the semantic rule. Your rules should generate sentences like the following:

(4) *a.* The drunk singer is cute.
b. Every drunk black cat is cute.

(*b*) Give the syntactic tree your grammar associates with (4*b*) along with its node-by-node translation on the model of the example on p. 326.

Exercise 7 Add to F_3 the following rule for possessives:

(1) NP → NP[+P + POSS] Nom

Here +P stands for "proper" or "pronominal," and +POSS stands for "possessive." The grammar of F_3 so augmented will generate sentences like the following:

(2) *a.* Sophia Loren likes her mother.
b. Pavarotti gives Sophia Loren's book to Bond.

Give the semantics corresponding to (1) in the form of a translation rule on the model of section 2. Such a translation rule should have the following form:

(3) If $\Delta = [\mathrm{NP}_i\ \mathrm{S}]$, where NP_i is of the form [NP Nom], then $\Delta' = \ldots$

Such a semantics will involve bringing in something like a possessor relation POS. POS(x, y) is a highly context-dependent relation that can mean, among other things, x belongs to y, x is assigned to y, or x was produced by y.

Illustrate your analysis by giving a node-by-node translation of (2*b*). It will help you if you find a meaning-preserving paraphrase of the possessive construction that uses more familiar determiners (*a*, *the*, or *every*).

6 Conclusions

In the present chapter we have familiarized ourselves with the logical syntax and semantics of the λ-operator, a very powerful device that enables one to define properties of indefinite complexity. One of the things that such an operator makes possible is doing semantics by means of translation into a logical calculus, which raises interesting questions concerning the notions of logical form and semantic representation. We have also seen how the λ-operator appears to be extremely useful in analyzing the structure of complex predicates in English, in particular, relative clauses, conjoined and disjoined VPs, and VP anaphors. We have considered a number of issues that these phenomena raise for the syntax-semantics interface.