15 Interpretations, Soundness, and Completeness for Predicate Logic

15–1. INTERPRETATIONS

In chapter 2 I introduced the idea of an interpretation for a predicate logic sentence, that is, of a case which determines the truth value for closed sentences of predicate logic. In the definition of chapter 2 I required that every object in the domain of an interpretation have at least one name. I included this requirement because with it I could give a simple and intuitive truth definition for existentially and universally quantified sentences: I said that an existentially quantified sentence is true in any interpretation just in case at least one of its substitution instances is true in the interpretation. And I said that a universally quantified sentence is true in an interpretation just in case all of its substitution instances are true in the interpretation.

Requiring every object to have a name may have been expedient for teaching fundamentals, but ultimately the requirement is unsatisfactory. Our system of logic should be able to deal with situations in which some objects go unnamed. So henceforth, by an interpretation for predicate logic, I will mean exactly what I meant in chapter 2, except that I will no longer require every object to have a name. I also will streamline the definition somewhat by counting atomic sentence letters as *Zero Place Predicates:*

> D20: An *Interpretation* consists of a nonempty domain of objects, a list of names, and a list of (zero place, one place, two place, and in general many place) predicates. The list of names may be empty, but there must be at least one predicate. For each name, the interpretation specifies the object in the domain which is named by that name; and for each predicate the interpretation specifies its truth value if it is a zero place predicate (an atomic sentence letter), or the objects in the domain of which the predicate is true if it is a one place predicate, or the ordered lists of objects of which the predicate is true if it is a two, three, or many place predicate. If a predicate is not true of an object or ordered list of objects, it is false of that object or list of objects.

This definition allows us to consider situations in which there are objects without names in the object language. But it makes hash of my definition of truth in an interpretation for quantified sentences.

Before we begin, precision requires a comment on notation. Remember that '(∃**u**)**P**(**u**)' is an expression of the metalanguage ranging over closed existentially quantified sentences, with **u** the existentially quantified variable. Ordinarily, **P**(**u**) will be an open sentence with **u** the only free variable, which is the way you should think of '**P**(**u**)' while getting an intuitive grasp of the material. But strictly speaking, '(∃**u**)**P**(**u**)' ranges over closed existentially quantified sentences, the **s**-substitution instances of which are **P**(**s**), the expressions formed by substituting **s** for all free occurrences of **u** in **P**(**u**)—**if there are any free occurrences of u**. This detail accommodates vacuously quantified sentences, such as '(∃x)A', as discussed in exercise 3–3.

To work toward new truth definitions for the quantifiers, let's think through what we want these definitions to do. Intuitively, (∃**u**)**P**(**u**) should be true in an interpretation iff there is some object in the domain of the interpretation of which the open sentence, **P**(**u**), is true. When all objects in the domain had names, we could express this condition simply by saying that there is at least one name, **s**, in the interpretation for which the substitution instance, **P**(**s**), is true in the interpretation. But now the object or objects in virtue of which (∃**u**)**P**(**u**) is true might have no names, so this strategy won't work.

We can get around this problem by appealing to the fact that, even if the interpretation we are considering does not include a name for the object we need, there will always be another interpretation which **does** have a name for this object and which is otherwise exactly the same.

In more detail, here is how the idea works. Suppose we have an interpretation, **I**, and a sentence (∃**u**)**P**(**u**). Intuitively speaking, (∃**u**)**P**(**u**) is true in **I** when **I** has an object, **o**, of which, intuitively speaking, the open sentence **P**(**u**) is true. We cannot say that (∃**u**)**P**(**u**) is true of **o** by saying that **o** has a name, **s**, in **I** such that **P**(**s**) is true in **I**. We are considering an example in which **o** has no name in **I**. But we get the same effect in this way: We consider a second interpretation, **I′**, which is **exactly like I**, except that in **I′** we assign **o** a name. We can always do this, because if **I**

is one interpretation, we get a second interpretation, **I′**, which has exactly the same domain of objects, the same list of predicates, the same specification of what is true of what, but which differs from **I** only by assigning the name **s** to object **o**.

We do also have to require that **s** not be a name which occurs in **(∃u)P(u)**. If, in going from **I** to **I′**, we move a name from one object to another, and this name occurs in **(∃u)P(u)**, we may disturb some other aspect of the truth conditions for **(∃u)P(u)**.

Some new terminology will help in transforming this intuitive idea into a precise definition:

> D21: $\mathbf{I}_s$ is an *s-Variant* of **I** iff $\mathbf{I}_s$ assigns the name **s** to some object in its domain and $\mathbf{I}_s$ differs from **I** at most by having name **s** or by assigning **s** to a different object.

With the help of the idea of an **s**-variant, we can say

> D22: **(∃u)P(U)** is true in interpretation **I** iff, for some name, **s**, which does not appear in **(∃u)P(u)**, there is an **s**-variant, $\mathbf{I}_s$, of **I** in which **P(s)** is true.

EXERCISE

15–1. Give an example of a sentence and an interpretation which shows that D22 would not work as intended if it did not include the requirement that **s** not appear in **(∃u)P(u)**.

The truth definition for the universal quantifier works in exactly the same way, except that we use 'all **s**-variants' instead of 'some **s**-variant'. We want to specify the conditions under which **(∀u)P(u)** is true in **I**. Intuitively, the condition is that **P(u)** be true of **all** objects in **I**. We capture this idea with the requirement that **P(s)** be true in **all** **s**-variants of **I**:

> D23: **(∀u)P(u)** is true in interpretation **I** iff, for some name, **s**, which does not appear in **(∀u)P(u)**, **P(s)** is true in all **s**-variants of **I**.

EXERCISE

15–2. Give an example of a sentence and an interpretation which shows that D23 would not work as intended if it did not include the requirement that **s** not appear in **(∀u)P(u)**.

I hope you will find these new truth definitions for quantifiers to have some plausibility. But they are a bit abstract and take some getting used to. The only way to become comfortable with them is to work with them. We can get the needed practice, and at the same time lay the groundwork for the next sections, by proving some basic lemmas.

Consider a predicate logic sentence, **X**, and an interpretation, **I**. Now consider some name which does not occur in **X**. If we reassign the name to some new object in the interpretation, this should make no difference to the truth value of **X** in **I**. **X** does not constrain the referent of the name in any way. The same thing goes for a predicate symbol not occurring in **X**. Intuitively, **X** and the unused predicate have no bearing on each other. So what the predicate is true of (or the truth value of a zero place predicate) should make no difference to the truth or falsity of **X**:

> L25: Let **X** be a sentence and **I** and **I′** two interpretations which have the same domain and which agree on all names and predicates which occur in **X**. Then **X** is true in **I** iff **X** is true in **I′**.

By 'agreeing on all names and predicates which occur in **X**', I mean that, for each name which appears in **X**, **I** and **I′** assign the same object to that name, and for each predicate appearing in **X**, **I** and **I′** specify the same truth value or the same collection of objects of which the predicate is true. For names and predicates not appearing in **X**, **I** and **I′** may make different assignments.

We prove L25 by induction on the number of connectives in **X**. For the basis case, consider an atomic **X** and an **I** and **I′** with the same domain which agree on all names and predicates in **X**. An interpretation explicitly provides the truth values in terms of the extensions of the used predicates and names (e.g., 'Pa' is true in **I** just in case the thing named 'a' is in the extension which **I** assigns to 'P'). Since **I** and **I′** agree on the predicates and names in **X**, they assign **X** the same truth value.

For the inductive case, assume, as inductive hypothesis, that L25 holds for all **X** with n or fewer connectives and all **I** and **I′** agreeing on **X**, as before. We must separately consider each of the connectives. For example, suppose that **X** has the form **Y&W**. Then **X** is true in **I** iff both **Y** and **W** are true in **I**. But since **Y** and **W** both have fewer connectives than **X**, we can apply the inductive hypothesis to conclude that **Y** is true in **I** iff **Y** is true in **I′**; and **W** is true in **I** iff **W** is true in **I′**. Finally, **Y** and **W** are both true in **I′** iff **X** (=**Y&W**) is true in **I′**, which is what we need to show in this part of the argument.

EXERCISE

15–3. Carry out the inductive step of the proof of L25 for the other sentence logic connectives, modeling your proof on the example just given for '&'.

Now assume that **X** has the form **(∃u)P(u)**. The ideas are not hard, but keeping everything straight can be confusing. So let's introduce some further terminology: For **I'** I will write **I(X)** to remind us that **I(X)** is an interpretation with the same domain as **I** and just like **I** so far as names and predicates in **X** are concerned, but differing arbitrarily from **I** on other predicates and names. In considering the case of **X = (∃u)P(u)**, instead of writing out **I((∃u)P(u))**, I will write just **I(P)**. Finally, I will write **I(P,s)** for an otherwise arbitrary interpretation agreeing with **I** on domain, on **P**, and on **s**.

So suppose that **(∃u)P(u)**, **I**, and **I(P)** have been given. Suppose that **I** makes **(∃u)P(u)** true. Definition D22 then tells us that there is a name, **s**, not appearing in **(∃u)P(u)**, and an **s**-variant of **I**, $\mathbf{I_s}$, where **P(s)** is true in $\mathbf{I_s}$. Now we change $\mathbf{I_s}$. We keep $\mathbf{I_s}$'s assignment of **s** and of all the names and predicates in **(∃u)P(u)**, and we change everything else to look just like **I(P)**. The resulting interpretation, **I(P,s)**, is an **s**-variant of **I(P)**. Furthermore, the inductive hypothesis applies to tell us that, since **P(s)** is true in $\mathbf{I_s}$, **P(s)** is true in **I(P,s)**. D22 applies to these facts to yield the conclusion that **(∃u)P(u)** is true in **I(P)**.

I have shown that if **(∃u)P(u)** is true in **I**, it is true in **I(P)**. But exactly the same argument works in the reverse—direction—if **I(P)** agrees with **I** on all vocabulary in **(∃u)P(u)**, then **I** agrees with **I(P)** on this vocabulary. So we may conclude that **(∃u)P(u)** is true in **I** iff it is true in **I(P)**, as was to be shown. (I did not use an iff in the chain of inferences in the previous paragraph because doing so makes it harder to keep clear about the existential quantifiers, 'there is an **s**' and 'there is an $\mathbf{I_s}$'. I will avoid certain 'iffs' in the proof of the next lemma for the same reason.)

EXERCISE

15–4. Carry out the inductive step of the proof of L25 for the universal quantifier.

Let's move on to another very intuitive fact, but one which is a bit tricky to prove. Consider a sentence of the form **R(s,t)**, a perhaps very complex sentence in which the names **s** and **t** may (but do not have to) occur. Let **I** be an interpretation in which **s** and **t** refer to the same object. Then it should not make any difference to the truth of **R(s,t)** in **I** if we replace any number of occurrences of **s** with occurrences of **t** or occurrences of **t** with occurrences of **s**. In **I**, **s** and **t** are just two different ways of referring to the same thing. **R(s,t)** says something about this thing, and how one refers to this thing should not make any difference to the truth of **R(s,t)** in **I**. (At this point it would be a good idea to review the discussion of extensional semantics in section 9–2.)

L26: Let **R(s,t)** be a closed sentence in which the names **s** and **t** may occur. Let **I** be an interpretation in which the names **s** and **t** refer to the same object. Let **R'(s,t)** arise by replacing any number of instances of **s** by **t** or instances of **t** by **s**. then **R(s,t)** is true in **I** iff **R'(s,t)** is true in **I**.

I have stipulated that **s** and **t** do not have to occur in **R(s,t)** to cover the important case in which **all** occurrences of **s** in a sentence **P(s)** get replaced by occurrences of **t**.

EXERCISE

15–5. Begin the proof of L26 by carrying out the basis step and the inductive step for the sentence logic connectives.

The complications in the inductive step for L26 call for writing it out in some detail. In what follows, take care to understand what I mean by '**r = s**'. '**r**' and '**s**' are metavariables over **names.** So '**r = s**' means that the name picked out by '**r**' is identical to the name picked out by '**s**', that is, that **r** and **s** are the same name. '**r = s**' does **not** mean the object referred to by the name picked out by '**r**' is the same as the object referred to by a different name picked out by '**s**'.

Now let's assume (inductive hypothesis) that L26 holds for all **R(s,t)** with n or fewer connectives. And let's consider the case of **R(s,t)** with the form **(∃u)Q(u,s,t)**. **R'(s,t)** is then the sentence **(∃u)Q'(u,s,t)**. Let interpretation **I** be given with names **s** and **t** having the same referent. In outline, the argument runs as follows:

(1) Suppose that **I** makes **(∃u)Q(u,s,t)** true. (Assumption)
(2) Then there is a name, **r**, and an **r**-variant, $\mathbf{I_r}$ of **I**, such that $\mathbf{I_r}$ makes **Q(r,s,t)** true. (By (1) and D22)
(3) Suppose that **r ≠ s** and **r ≠ t**. (Assumption, to be discharged)
(4) Then $\mathbf{I_r}$ makes **Q'(r,s,t)** true. (By the inductive hypothesis applied to (2) and (3))
(5) Then **I** makes **(∃u)Q'(u,s,t)**. (By D22 applied to (4))

I want to be sure you understand step (4) and the role of step (3). First, you might have thought that D22 guarantees (3). But that happens only if both **s** and **t** actually occur in **(∃u)Q(u,s,t)**. Since we want out proof to cover, for example, a sentence in which just **t** occurs and in which we replace all occurrences of **t** with occurrences of **s**, we have allowed that **s** and **t** don't have to occur. Next, remember that to apply the inductive hypothesis to switch around the names **s** and **t**, we need to be considering an interpretation in which **s** and **t** both refer to the same object. By assumption, **I** is such an interpretation. But in step (4) we need this to be

true of $\mathbf{I}_r$. If $\mathbf{r} \neq \mathbf{s}$ and $\mathbf{r} \neq \mathbf{t}$, we're OK. According to D22, $\mathbf{I}_r$ arises from **I** by at most reassigning **r** to a new referent. When $\mathbf{r} \neq \mathbf{s}$ and $\mathbf{r} \neq \mathbf{t}$, **s** and **t** still have their mutual referent, so the inductive hypothesis can be applied.

To get ready to discharge the assumption (3), let's see what can go wrong if (3) fails. Let's suppose that $\mathbf{r} = \mathbf{s}$. In this case, when we apply D22 to make $\mathbf{I}_r$ out of **I**, we might have the situation pictured for $\mathbf{I}_r$ in figure 15–1.

In **I**, **s** and **t** both refer to the object $\mathbf{o}_t$. We apply D22, which says that there is an **r**-variant, $\mathbf{I}_r$, of **I**, differing at most from **I** by assigning a new referent, which I'm calling '$\mathbf{o}_r$', to **r** ($\mathbf{o}_r$ is an object which makes the existential quantification true). But if $\mathbf{r} = \mathbf{s}$, this means assigning **r**, that is, **s**, to the object $\mathbf{o}_r$, which in general will be distinct from $\mathbf{o}_t$. So in $\mathbf{I}_r$ we may not have available the condition that **s** and **t** have the same referent, the condition needed to apply the inductive hypothesis.

To get around this difficulty I will argue by cases. Case 1: Neither **s** nor **t** actually occurs in $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},\mathbf{s},\mathbf{t})$. Then there is nothing to prove, since there are no occurrences of **s** and **t** to switch around. Case 2: **s** and **t** both occur in $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},\mathbf{s},\mathbf{t})$. D22 requires that **r** not occur in $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},\mathbf{s},\mathbf{t})$. So in this case $\mathbf{r} \neq \mathbf{s}$ and $\mathbf{r} \neq \mathbf{t}$, we have assumption (3) available, and the proof (1)–(5) can proceed.

Case 3: **t** but not **s** actually occurs in $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},\mathbf{s},\mathbf{t})$. (The case in which **s** but not **t** occurs is the same.) To remind us that **s** occurs vacuously, I will put parentheses around **s**, like this: $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},(\mathbf{s}),\mathbf{t})$. If, in this case, **r** happens by luck to be distinct from **s**, the proof (1)–(5) applies. So I will also assume that $\mathbf{r} = \mathbf{s}$. In this case we have the situation for $\mathbf{I}_r$ pictured in figure 15–1, and the inductive hypothesis will not apply because **s** and **t** no longer have the same referent. In addition, we won't be able to apply

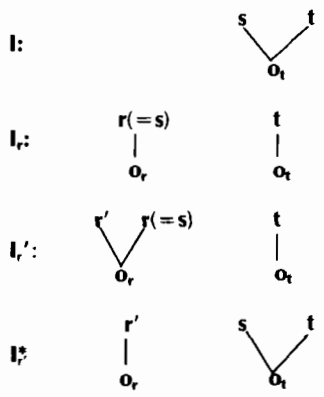


Figure 15–1

D22 in step (5). When $\mathbf{r} = \mathbf{s}$, **r**, that is, **s**, will get put in for occurrences of **t** when we exchange $\mathbf{Q}(\mathbf{r},(\mathbf{s}),\mathbf{t})$ for $\mathbf{Q}'(\mathbf{r},(\mathbf{s}),\mathbf{t})$. Then when we try to apply D22 to reform the existential quantification, the **u** will get put into the wrong places.

To resolve these difficulties, I must accomplish two things. I must show that I can pick another name, **r'**, with $\mathbf{r}' \neq \mathbf{s}$ and $\neq \mathbf{t}$, and assign **r'** to $\mathbf{o}_r$. Then I must **reassign** **s** as a name of $\mathbf{o}_t$. If I do these two things, then **s** and **t** will again have the same referent, so that I can apply the inductive hypothesis in step (4); and I will again be using a name, $\mathbf{r}' \neq \mathbf{s}$ and $\neq \mathbf{t}$, so that D22 will unproblematically apply in step (5).

Once this problem is clearly explained it is easy to solve, with the help of lemma L25. I pick a new name, **r'**, not occurring in $\mathbf{Q}(\mathbf{r},(\mathbf{s}),\mathbf{t})$, $\neq \mathbf{s}$ and $\neq \mathbf{t}$. L25 tells us that $\mathbf{Q}(\mathbf{r},(\mathbf{s}),\mathbf{t})$ has the same truth value in a new interpretation, $\mathbf{I}_{r'}$, that it had in $\mathbf{I}_r$, where $\mathbf{I}_{r'}$ is just like $\mathbf{I}_r$ except that **r'** has been assigned as an additional name of $\mathbf{o}_r$. Next I apply the inductive hypothesis to $\mathbf{Q}(\mathbf{r},(\mathbf{s}),\mathbf{t})$ and the interpretation $\mathbf{I}_{r'}$. In $\mathbf{I}_{r'}$, **r'** and **r** (that is, **s**) both name $\mathbf{o}_r$. So the inductive hypothesis allows me to replace all occurrences of **r** with **r'**. I now have $\mathbf{Q}(\mathbf{r}',(\mathbf{s}),\mathbf{t})$ true in $\mathbf{I}_{r'}$, with **s** not actually occurring in $\mathbf{Q}(\mathbf{r}',(\mathbf{s}),\mathbf{t})$. Consequently, I can again apply L25 to tell me that $\mathbf{Q}(\mathbf{r}',(\mathbf{s}),\mathbf{t})$ is also true in $\mathbf{I}_r^*$, an interpretation just like $\mathbf{I}_{r'}$ except that **s** has been reassigned to $\mathbf{o}_t$. At this point $\mathbf{I}_r^*$ is an **r'**-variant of **I**, $\mathbf{r}' \neq \mathbf{s}$ and $\neq \mathbf{t}$, and **s** and **t** are both referents of $\mathbf{o}_t$ so that I can carry out steps (4) and (5) of the foregoing proof using $\mathbf{I}_r^*$, instead of $\mathbf{I}_{r'}$.

We are almost done. I have shown that if **I** makes $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},\mathbf{s},\mathbf{t})$ true, then **I** makes $(\exists \mathbf{u})\mathbf{Q}'(\mathbf{u},\mathbf{s},\mathbf{t})$ true. But the argument works in exactly the same way in the opposite direction. So we have shown that **I** makes $(\exists \mathbf{u})\mathbf{Q}(\mathbf{u},\mathbf{s},\mathbf{t})$ true iff it makes $(\exists \mathbf{u})\mathbf{Q}'(\mathbf{u},\mathbf{s},\mathbf{t})$ true, completing this part of the proof of L26.

EXERCISES

15–6. In a more advanced logic text, this sort of informal proof would be written up much more briefly. I have spelled it out in some detail to help you learn how to read and study such a proof. To further practice study of an informal proof and to appreciate better how complicated it really is, formalize the proof as a natural deduction. Use '$\mathbf{Mod}(\mathbf{I},\mathbf{X})$' for '**X** is true in **I**', '$(\exists \mathbf{r})$' for 'There is a name, **r**', '$(\exists \mathbf{I}_r)$' for 'There is an **r**-variant, $\mathbf{I}_r$, of **I**', and so on. I suggest that you formalize the initial proof (1)–(5), with the undischarged assumption of step (3), being sure to make explicit the tacit appeal to ∃E. Then fill in the full argument explained in the discussion which follows (1)–(5). The most efficient natural deduction may have a significantly different organization than the informal presentation,

which was designed to help you see what is going on as opposed to presenting the argument in as few steps as possible.

Students of truth trees may also have fun doing this argument as a truth tree proof, although this is less helpful in exposing the structure of the informal argument in English.

15–7. Carry out the inductive step of the proof of L26 for universally quantified sentences. You may do this most efficiently by commenting on how to modify the proof for the case of existentially quantified sentences.

Now that I have shown you how to proceed with this kind of argument, I am going to ask you to prove the rest of the lemmas we will need. When not otherwise specified, **P(u)** can be any open sentence with **u** the only free variable, **I** any interpretation, and so on.

Lemmas L27 and L28 show that the truth definitions for the quantifiers are equivalent to conditions which, superficially, look stronger than the definitions:

L27: Let **s** be **any** name not appearing in **(∃u)P(u)**. Then **Mod[I,(∃u)P(u)]** iff there is an **s**-variant, $\mathbf{I_s}$, of **I** such that **Mod[$\mathbf{I_s}$, P(s)]**.

L28: Let **s** be **any** name not appearing in **(∀u)P(u)**. Then **Mod[I,(∀u)P(u)]** iff **Mod[$\mathbf{I_s}$,P(s)]** for all **s**-variants, $\mathbf{I_s}$, of **I**.

EXERCISE

15–8. Prove L27 and L28. Apply L25 and L26 to D22 and D23. You will not need to do an induction.

L29: **~(∀u)P(u)** is logically equivalent to **(∃u)~P(u)** and **~(∃u)P(u)** is logically equivalent to **(∀u)~P(u)**.

EXERCISE

15–9. Prove L29. Remember that logical equivalence is the semantic notion of having the same truth value in all interpretations. You will not need to use induction. Instead, simply apply L27 and L28.

When you have finished your proof of L29, look it over and find the places at which you used, as informal logical principles applied in the metalanguage, just the negated quantifier rules which you were proving as generalizations about the object language! It is a noteworthy, and perhaps disturbing, fact that we cannot prove anything about the object language formulation of logic without assuming logical principles at least as strong in the metalanguage. What, then, do we gain in the process? Precision and clarity.

L30: Suppose that **Mod[I,P(s)]**. Then **Mod[I,(∃u)P(u,s)]**, where **P(u,s)** arises from **P(s)** by substituting **u** for any number of occurrences of **s** in **P(s)**.

L31: Suppose that **Mod[I,(∀u)P(u)]**. Let **I′** differ from **I** only in assignment of names not occurring in **(∀u)Pu,** and let **s** be any name in **I′**. Then **Mod[I′,P(s)]**.

Note that in L31, **s** may be a name appearing in **(∀u)P(u)**. L31 is a generalization of the principle that all substitution instances of a universally quantified sentence are true, a generalization we will need in the following sections.

EXERCISES

15–10. Prove L30. You will use L26 and D22 and no induction.

15–11. Prove L31, using L25, L26, and L28. The fact that **s** may appear in **(∀u)P(u)** may give you trouble in this problem. The trick is not to use the name **s** for the **s**-variant in D23. Use some other name, **t**, which does not appear in **(∀u)P(u)** and then apply L25 and L26 to **s** and **t**.

L32: Let **I** be an interpretation in which every object in its domain has a name. Then

a) **Mod[I,(∃u)P(u)]** iff **Mod[I,P(s)]** for some name, **s**, that appears in **I**.
b) **Mod[I,(∀u)P(u)]** iff **Mod[I,P(s)]** for all names, **s**, that appear in **I**.

L32 simply says that the truth definitions for quantifiers given in chapter 2 work in the special case in which all objects in an interpretation's domain have names.

EXERCISE

15–12. Using any prior definitions and lemmas from this section that you need, prove L32.

We are now ready to extend our previous proofs of soundness and completeness for sentence logic to predicate logic. Most of the real work has been done in the lemmas of this section and in Koenig's lemma from

chapter 14. I am only going to outline the proofs and ask you to fill in the details. In the next three sections I will only treat predicate logic without identity or function symbols, and I will treat only finite sets of sentences in the completeness proofs.

15–2. SOUNDNESS AND COMPLETENESS FOR TREES

When we extend trees for sentence logic to predicate logic, we add four new rules: ∃, ~∃, ∀, and ~∀. Roughly speaking, what we need to do is to check that these rules are downwardly and upwardly correct. There is, however, a complication: infinite trees.

Before going further, please review section 8–4. In sentence logic every tree ends, including all open trees. That is because we need to work on each sentence only once, and when working on a sentence the sentences on the output list are all shorter than the input sentence. But in predicate logic we may have to work on sentences of the form **(∀u)(∃v)R(u,v)** more than once. When we instantiate **(∀u)(∃v)R(u,v)** with a name, **s**, we get an existentially quantified sentence, **(∃v)R(s,v)**. When we apply ∃ to this sentence, we **must** use a new name, **t**, which we must then substitute back into **(∀u)(∃v)R(u,v)**, producing another existentially quantified sentence, which will produce another new name, and so on.

The overall tree strategy still works as before: We make longer sentences true by making shorter sentences true, until we get to minimal sentences which describe an interpretation. The process may now go on forever, but we can still think of infinite open paths as describing interpretations in which all sentences on the paths are true.

Because trees can be infinite, we need to reconsider what is involved in a finished tree. We do not need to revise our definition, D9, but we do need to make sure that if a tree does not close in a finite number of steps that it can be finished in the sense given by D9. That is, we must make sure that there is some systematic way of applying the rules which guarantees that, for each sentence to which a rule can be applied, eventually the rule is applied.

Here's a system which supplies the guarantee. We segment our work on a tree into stages. At stage n we work only on sentences that appear on lines 1 through n. Stage n continues until all sentences on lines 1 through n which can be checked have been checked and until all names occurring in lines 1 through n have been substituted into all universally quantified sentences occurring in lines 1 through n. Of course, at the end of stage n, the tree may have grown to many more than n lines. But that does not matter. Every checkable sentence occurs by some line, n, and so will eventually get checked by this process, and every name and every universally quantified sentence occurs by some line n, so every universally quantified sentence will eventually be instantiated by every name. Of course, this system is not efficient. But efficiency is not now the point. We want to show that there is a system which is guaranteed not to leave anything out.

The next point to establish is that if a tree is infinite it has an infinite open branch. Koenig's lemma tells us that if a tree is infinite, it has an infinite branch, and since closed branches are finite, this infinite branch must be open.

Open branches, infinite or finite, describe interpretations in pretty much the way they do for sentence logic. Given an open branch, collect all the names that occur on the branch and set up a domain of objects, each one named by one of the names on the branch, with no two names assigned to the same object. Then let the minimal sentences on the branch specify what is true of what. Atomic sentence letters are treated as in sentence logic. If an atomic sentence of the form **P(s)** appears on the branch, in the branch's interpretation **P** is true of **s**. If an atomic sentence of the form **~R(s,t)** appears, then in the branch's interpretation **R** is false of the pair of objects named by **s** and **t** (in that order). And so on. The minimal sentences will generally fail to specify all atomic facts. The unspecified facts may be filled in arbitrarily.

For sentence logic we formulated rule correctness in terms of **any** interpretation: Any interpretation which makes an input sentence true makes at least one output list true. And any interpretation which makes an output list true makes the input sentence true. This won't work for quantified sentences.

For upward correctness of the ∀ rule, consider some sentence, **(∀u)P(u)**, and some interpretation, **I**, in which there are more objects than are named on an open branch. Even if all of the output sentences of the ∀ rule—that is, even if all of the substitution instances of **(∀u)P(u)** which appear on this branch—are true in **I**, **(∀u)P(u)** might not be true in **I**. To be true in **I**, **(∀u)P(u)** must be true for **all** objects in **I**, whether the object concerned has a name or not.

For downward correctness we need the following: Given an interpretation in which the first n lines of a branch are true, there is an interpretation which makes true all of these sentences as well as the sentences in an output list resulting from applying a rule. But for the ∃ rule, not just any interpretation in which the first n lines, including **(∃u)P(u)**, are true will serve. Such an interpretation might not have a name for an object which makes **(∃u)P(u)** true. Worse, the interpretation might have such a name but the resulting substitution instance might conflict with another sentence already on the branch.

This last problem is what necessitated the new name rule, and it is essential that you understand how that requirement fits in here. Suppose that our branch already has '(∃x)Bx' and '~Ba' and that the interpreta-

tion, **I**, which makes these two sentences true has just one name, '**a**', and two objects, the first, named by 'a', which is not B and the second, which has no name in **I** and is B. This **I** is a consistent interpretation for '(∃x)Bx' and '~Ba', but we cannot use it in forming a substitution instance which shows '(∃x)Bx' to be true. We must extend or change our interpretation by assigning a new name, 'b', to the unnamed object. Then the truth of '(∃x)Bx' is made explicit by including 'Bb' on the branch.

The new name feature of the ∃ rule ensures that we always proceed in the way just described. When it comes time to describe downward correctness of the ∃ rule, the downward correctness must be given a corresponding description. As in the last example, the **I** which makes the initial sentences on the branch true may not have the required new name. Or **I** may have the name but, since the name does not occur in any of the sentences so far considered on the branch, the name could refer to the wrong object. (Think of lemma 25 in making sure you understand this last point.) For lack of the right name referring to the right object, the **I** which makes true the first n sentences on a branch may not also make true the substitution instance which comes by applying the ∃ rule with its new name requirement. But there will always be an **s**-variant of **I**, $\mathbf{I_s}$, resulting by assigning the new name **s** to the right object, which will make true **(∃u)P(u)**'s substitution instance, **P(s)**. Since **s** is new to the branch, lemma 25 guarantees that all the prior sentences in the branch will still be true in $\mathbf{I_s}$.

The foregoing remarks should motivate the following revisions of D15 and D16:

> D15′: A tree method rule is *Downwardly Correct* iff it meets the following condition for all interpretations, **I**, and all line numbers, n: Suppose that **I** is an interpretation which makes true all sentences along a branch from lines 1 through n. Suppose that the input sentence for the rule lies on this branch, on one of the lines 1 through n, and the sentences on the output lists lie on the lines immediately succeeding n. Then there is an **s**-variant of **I** which makes true all of the sentences on the original branch, lines 1 through n, and also all of the sentences on at least one of the output lists.
>
> D16′: A tree method rule is *Upwardly Correct* iff in any interpretation, **I**, which is described by an open branch, if all the sentences on an output list on that branch are true in **I**, then the input sentence is true in **I**.

Note that upward correctness concerns **only** interpretations which are described by the open branch in question.

Before checking rule correctness, we need to clarify what is to count as the output list for an application of the ∀ rule. For upward correctness, the output list resulting when ∀ is applied to **(∀u)P(u)** includes all the substitution instances of **(∀u)P(u)** on the finished branch. For downward correctness the output list includes only those substitution instances on the branch as it exists just after the ∀ rule is applied to **(∀u)P(u)** but before any further rules are applied.

You can now proceed to check downward and upward correctness of the quantifier rules.

EXERCISES

15–13. Using lemma L29, show that the rules ~∃ and ~∀ are downwardly and upwardly correct according to D15′ and D16′ (though, for these two rules, the difference with D15 and D16 is inessential).

15–14. Prove that the ∃ rule is upwardly correct. You only need apply definition D22.

15–15. Prove that the ∀ rule is upwardly correct. You need to apply lemma L32.

15–16. Prove that the ∃ rule is downwardly correct. You need lemmas L25 and L27. Note carefully the role of the new name requirement in your proof.

15–17. Prove that the ∀ rule is downwardly correct. You need lemma L31. Don't forget to treat the case in which ∀ applies to a sentence on a branch with no names. This case will require L25.

We have now done all the real work in proving downward and upward adequacy:

> T10: The truth tree method for predicate logic is downwardly adequate.
>
> T11: The truth tree method for predicate logic is upwardly adequate.

Given the revised definitions of upward rule correctness, the proof of upward adequacy works pretty much as it does for sentence logic. Downward adequacy requires some change, in ways which I have already indicated. Suppose that an initial set of sentences has a model. For sentence logic we showed that each time we applied a rule there is at least one extension of the initial segment of a branch all the sentences of which are true in the original model. Now we show instead that each time we apply a rule there is at least one extension of the initial segment of a branch all the sentences of which are true in an **s**-variant of the model for the prior branch segment. D15′ has been designed to make the inductive proof of this statement straightforward.

EXERCISES

15–18. Prove downward adequacy for predicate logic trees.

15–19. Prove upward adequacy for predicate logic trees. To extend the proof of section 12–2, you will need to revise the definition of

'length of a sentence'. The natural alternative is to let the length of a predicate logic sentence be the number of predicates and connectives. But on this definition the input and output sentences of the $\sim\exists$ and $\sim\forall$ rules have the same length. With a little care you can still do the induction with this definition. Or you can define length by letting an initial negation followed by a quantifier count as three units of length and letting each occurrence of '$\equiv$' count as two units.

T10 and T11, downward and upward adequacy, immediately give

T12: The truth tree method for predicate logic is sound.

and

T13: The truth tree method for predicate logic is complete.

in exactly the way they do for sentence logic.

15–3. SOUNDNESS FOR PREDICATE LOGIC DERIVATIONS

To extend the proof for sentence logic, we need to prove rule soundness for the four new predicate logic rules. Two are easy applications of definitions and lemmas given in section 15–1:

L33 (Soundness for $\exists$I): If $Z\models\mathbf{P(s)}$, then $Z\models(\exists\mathbf{u})\mathbf{P(u,s)}$,where $(\exists\mathbf{u})\mathbf{P(u,s)}$ is an existential generalization of $\mathbf{P(s)}$, that is, $\mathbf{P(u,s)}$ results from $\mathbf{P(s)}$ by replacing any number of occurrences of $\mathbf{s}$ with $\mathbf{u}$.

L34 (Soundness for $\forall$E): If $Z\models(\forall\mathbf{u})\mathbf{P(u)}$, then $Z\models\mathbf{P(s)}$, where $\mathbf{P(s)}$ is a substitution instance of $(\forall\mathbf{u})\mathbf{P(u)}$, that is, $\mathbf{s}$ is substituted for all free occurrences of $\mathbf{u}$ in $\mathbf{P(u)}$.

EXERCISES

15–20. Apply lemma L30 to prove lemma L33.
15–21. Apply lemma L31 to prove lemma L34.

Let's look at $\forall$I in a bit more detail. We want to prove

L35 (Soundness for $\forall$I): Assume that the name $\mathbf{s}$ does not occur in $\mathbf{Z}$ or in $(\forall\mathbf{u})\mathbf{P(u)}$. On this assumption, if $Z\models\mathbf{P(s)}$, then $Z\models(\forall\mathbf{u})\mathbf{P(u)}$, where $(\forall\mathbf{u})\mathbf{P(u)}$ is the universal generalization of $\mathbf{P(s)}$, that is, $\mathbf{P(u)}$ results by replacing **all** occurrences of $\mathbf{s}$ in $\mathbf{P(s)}$ with $\mathbf{u}$.

Let's consider an arbitrary interpretation, $\mathbf{I}$, in which all the sentences in $\mathbf{Z}$ are true. What will it take for $(\forall\mathbf{u})\mathbf{P(u)}$ to be true also in $\mathbf{I}$? Lemma L28 tells us that given any name, $\mathbf{s}$, not appearing in $(\forall\mathbf{u})\mathbf{P(u)}$, we need only show that $\mathbf{P(s)}$ is true in all $\mathbf{s}$-variants of $\mathbf{I}$. What we need to do is squeeze the conclusion that $\mathbf{P(s)}$ is true in all the $\mathbf{s}$-variants of $\mathbf{I}$ out of the assumption that $Z\models\mathbf{P(s)}$ and the hypothesis that $\mathbf{Mod(I,Z)}$.

But this is easy. The assumption that $\mathbf{s}$ does not occur in $\mathbf{Z}$ allows us to apply lemma L25 as follows: $\mathbf{I}$ is a model for $\mathbf{Z}$. Since $\mathbf{s}$ does not occur in $\mathbf{Z}$, L25 tells us that any $\mathbf{s}$-variant of $\mathbf{I}$ is also a model of $\mathbf{Z}$. Then the assumption that $Z\models\mathbf{P(s)}$ tells us that any $\mathbf{s}$-variant of $\mathbf{I}$ makes $\mathbf{P(s)}$ true.

You should carefully note the two restrictions which play crucial roles in this demonstration. In order to apply lemma L25, $\mathbf{s}$ must not appear in $\mathbf{Z}$. Also, in order to apply lemma L28, $\mathbf{s}$ must not appear in $(\forall\mathbf{u})\mathbf{P(u)}$. The latter restriction is encoded in the $\forall$I rule by requiring that $(\forall\mathbf{u})\mathbf{P(u)}$ be the universal generalization of $\mathbf{P(s)}$.

In a similar way, the restrictions built in the $\exists$E rule play a pivotal role in proving

L36 (Soundness for $\exists$E): Assume that $\mathbf{s}$ does not appear in $\mathbf{Z}$, in $(\exists\mathbf{u})\mathbf{P(u)}$, or in $\mathbf{X}$. Then if $Z\cup\{(\exists\mathbf{u})\mathbf{P(u)},\mathbf{P(s)}\}\models\mathbf{X}$, then $Z\cup\{(\exists\mathbf{u})\mathbf{P(u)}\}\models\mathbf{X}$.

You will immediately want to know why the restrictions stated in L36 are not the same as the restriction I required of the $\exists$E rule, that $\mathbf{s}$ be an isolated name. If you look back at section 5–6, you will remember my commenting that requiring $\mathbf{s}$ to be an isolated name involves three more specific requirements, and that other texts state the $\exists$E rule with these three alternative requirements. These three requirements are the ones which appear in the assumption of L36. Requiring that $\mathbf{s}$ be an isolated name is a (superficially) stronger requirement from which the other three follow. Since we are proving soundness, if we carry out the proof for a weaker requirement on a rule, we will have proved it for any stronger requirement. You can see this immediately by noting that if we succeed in proving L36, we will have proved any reformulation of L36 in which the assumption (which states the requirement) is stronger.

Of course, by making the requirement for applying a rule stronger (by making the rule harder to apply), we might spoil completeness—we might make it too hard to carry out proofs so that some valid arguments would have no corresponding proofs. But when we get to completeness, we will check that we do not get into that problem.

Let's turn to proving L36. The strategy is like the one we used in proving L35, but a bit more involved. Assume that $\mathbf{I}$ is a model for $\mathbf{Z}$ and $(\exists\mathbf{u})\mathbf{P(u)}$. Since $\mathbf{s}$ does not appear in $(\exists\mathbf{u})\mathbf{P(u)}$, there is an $\mathbf{s}$-variant, $\mathbf{I_s}$ of $\mathbf{I}$, such that $\mathbf{P(s)}$ is true in $\mathbf{I_s}$. Since $\mathbf{s}$ does not appear in $(\exists\mathbf{u})\mathbf{P(u)}$ or in $\mathbf{Z}$, and since $\mathbf{I}$ and $\mathbf{I_s}$ differ only as to $\mathbf{s}$, lemma L25 tells us that $(\exists\mathbf{u})\mathbf{P(u)}$ and

Z are also true in $\mathbf{I}_s$. The hypothesis, that $\mathbf{Z} \cup \{(\exists \mathbf{u})\mathbf{P}(\mathbf{u}), \mathbf{P}(\mathbf{s})\} \nvdash \mathbf{X}$, then tells us that **X** is true in $\mathbf{I}_s$. Finally, since **s** is assumed not to appear in **X** and **I** and $\mathbf{I}_s$ differ only as to **s**, lemma L25 again applies to tell us that **X** is true in **I**.

The soundness of the quantifier rules immediately gives us

> T14 (Soundness for predicate logic derivations): For any set of sentences, **Z**, and sentence, **X**, if $\mathbf{Z} \vdash \mathbf{X}$, then $\mathbf{Z} \models \mathbf{X}$.

The proof is a trivial extension of the proof for sentence logic, but to fix the ideas you should carry out this extension.

EXERCISE

15–22. Prove T14. You only need to extend the inductive step in the proof of T5 to cover the cases of the four quantifier rules.

15–4. COMPLETENESS FOR PREDICATE LOGIC DERIVATIONS

For completeness, we also follow the same overall strategy as we did for sentence logic. Starting with an initial tableau of sentences, we generate a new tableau the sentences of which make the sentences on the original tableau true. The sentences on the generated tableau are, on the whole, shorter than on the generating tableau. Roughly speaking, we eventually get down to minimal sentences which characterize an interpretation on which all the sentences of ancestor tableaux are true. But there will be some new wrinkles.

We have to say how quantified and negated quantified sentences will be treated on a tableau. For negated quantified sentences, we apply the rules of logical equivalence for negated quantifiers, pushing the negation sign through the quantifier and switching the quantifier. That will leave us with only quantified sentences, with no negation signs in front, with which we have to deal.

We will make a universally quantified sentence true by making all its substitution instances true. We will make an existentially quantified sentence true by making one substitution instance true. But we will have to make this substitution instance the assumption of a new subderivation so that we will be able to apply the ∃E rule to contradictions to get 'A&~A' as the final conclusion of the outermost derivation.

These ideas get incorporated by extending the rules for sequential and branching generation:

> R2′ *Sequential generation:* Extend the statement of the rule with the following steps (to be applied before the instruction to reiterate remaining sentences).
>
> a) If a sentence of the form $\sim(\exists \mathbf{u})\mathbf{P}(\mathbf{u})$ appears on the generating tableau, enter $(\forall \mathbf{u})\sim\mathbf{P}(\mathbf{u})$ on the generated tableau.
> b) If a sentence of the form $\sim(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ appears on the generating tableau, enter $(\exists \mathbf{u})\sim\mathbf{P}(\mathbf{u})$ on the generated tableau.
> c) If a sentence of the form $(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ occurs on the generating tableau, enter on the generated tableau all the substitution instances formed with names which appear on the generating tableau. If no names appear on the generating tableau, pick one name arbitrarily and use it to form a substitution instance entered on the generated tableau. **Also** reiterate $(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ on the generated tableau.

We must reiterate $(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ on the generated tableau because, as you will soon see, new names can arise on later tableaux. These new names must be substituted into $(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ to make $(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ true for all its substitution instances. So we must carry $(\forall \mathbf{u})\mathbf{P}(\mathbf{u})$ along on each tableau to have it for forming substitution instances any time that a new name arises.

> R3′ *Branching generation:* If any sentence of the form $\mathbf{X} \vee \mathbf{Y}$ occurs on the generating tableau, apply R3 exactly as stated. If no $\mathbf{X} \vee \mathbf{Y}$ occurs but there is a sentence of the form $(\exists \mathbf{u})\mathbf{P}(\mathbf{u})$ on the generating tableau, pick a **New Name,** that is, a name which does not appear anywhere on the generating tableau. Use the new name to form a substitution instance of $(\exists \mathbf{u})\mathbf{P}(\mathbf{u})$, and use this substitution instance as the assumption starting a new subderivation. Reiterate all other sentences on the generating tableau in the subderivation to complete the generated tableau, just as in R3.

As students of the tree method already know, these rules create a problem. Suppose that a sentence of the form $(\forall \mathbf{u})(\exists \mathbf{v})\mathbf{R}(\mathbf{u},\mathbf{v})$ appears on a tableau. R2′ tells us to enter at least one substitution instance, $(\exists \mathbf{v})\mathbf{R}(\mathbf{s},\mathbf{v})$, on the next tableau and to reiterate $(\forall \mathbf{u})(\exists \mathbf{v})\mathbf{R}(\mathbf{u},\mathbf{v})$ itself. R3′ will then tell us to start a new subderivation with $\mathbf{R}(\mathbf{s},\mathbf{t})$, **t** a new name. Of course, $(\forall \mathbf{u})(\exists \mathbf{v})\mathbf{R}(\mathbf{u},\mathbf{v})$ also gets reiterated onto the subderivation. But now we will have to do the same thing all over again. The new name, **t**, will have to go into $(\forall \mathbf{u})(\exists \mathbf{v})\mathbf{R}(\mathbf{u},\mathbf{v})$, giving a new existentially quantified sentence, $(\exists \mathbf{v})\mathbf{R}(\mathbf{t},\mathbf{v})$, which will call for a new subderivation with yet another new name, which will have to go back into the reiterated $(\forall \mathbf{u})(\exists \mathbf{v})\mathbf{R}(\mathbf{u},\mathbf{v})$. We are off and running in a chain of subderivations that will never end.

A first impulse is to wonder if the generation rules couldn't be written better, so as to avoid this problem. They can be written so as to avoid exactly this form of the problem, but it turns out that no matter how the rules are written, some problem with essentially the same import will arise. Indeed, proving this is a further important fact about logic.

Here is an overview of what the problem involves. The semantic tableau procedure provides a mechanical method for searching for a derivation

which establishes the validity of a given argument, or equivalently, a mechanical method for searching for an interpretation of a given finite set of sentences. In sentence logic, the method is guaranteed to terminate. A method which thus terminates is guaranteed to give a definite yes or no answer to the original question ('Is the argument valid?' or 'Is the initial set of sentences consistent?'). Such a method, guaranteed eventually to turn up a yes or no answer, is called a *Decision Procedure.*

Now, here is the general form of our current problem. Given an exceedingly plausible assumption about what will count as a mechanical decision procedure, one can prove that there is no decision procedure for predicate logic. In our formulation we fail to get a decision procedure because we may get an infinite sequence of sub-sub . . . -sub-derivations. If our tableau procedure has failed to close at some stage, we may not be able to tell for sure whether that is because we just haven't pursued it far enough, or because it will go on forever. This is not just a weakness of our rules. One can prove that any sound and complete system of predicate logic will suffer in the same way. Roughly speaking, the problem arises from the requirement on the ∃E rule, which we must have in order to ensure soundness.

Since there is no point in searching for better rules, we will have to see what we can make of our R2′ and R3′ in fashioning a completeness proof.

Consider a set of sentences, for example, just the sentence **(∀u)(∃v)R(u,v)** for which our tableau procedure generates an infinite sequence of tableaux. We will need the fact that we can then, so to speak, draw an unending path through the nested sequence of subderivations. Koenig's lemma assures us that we can always do so. Refer back to the tree structure at the beginning of chapter 14 and imagine that each node represents a subderivation, beginning with the outermost derivation at the top node. Moving from one node to two nodes beneath represents the process of starting two new subderivations by working on a sentence of the form **XvY.** When we start one new subderivation by working on a sentence of the form **(∃u)P(u)**, we start one new node, that is, a "branch" with one rather than two new forks. When a subderivation closes, the corresponding path on the tree structure closes. Koenig's lemma tells us that if such a tree structure is infinite, then there is an infinite open path through the tree.

We now know that if a tableau derivation does not close (is infinite or does not have all its terminal tableaux closed) then there is an open path of subderivations through the derivation. The path might be finite or it might be infinite. Each such path provides an interpretation, which we will again call a *Terminal Interpretation.* But we want to characterize the idea of a terminal interpretation so that it will work for infinite as well as finite cases. Since an infinite path through a derivation has no terminal tableau, we cannot let the terminal interpretation simply be one provided by the terminal tableau.

Here's the recipe for the terminal interpretation represented by an infinite path. Collect all the names that occur on the path, and set up a domain of objects, each one named by one of the names on the path, with no two names assigned to the same object. Then look at all the minimal sentences which appear on the path. If an atomic sentence letter appears, the interpretation will make it true. If an atomic sentence letter appears negated, the interpretation will make the atomic sentence letter false. If an atomic sentence of the form **P(s)** appears, the interpretation will make the predicate **P** true of the object named by **s**. Similarly, if **~P(s)** appears, the interpretation will make **P** false of the object named by **s**. Two and more place predicates are treated similarly. If this recipe fails to specify all the atomic facts of the interpretation, fill in the missing facts arbitrarily. In sum

> D24: A *Terminal Interpretation* represented by an open path has as its names all the names which occur on the path and as its domain a set of objects, each named by exactly one of the names. The interpretation assigns truth values to atomic sentence letters and determines which predicates are true of which objects (pairs of objects, and so on) as described by the minimal sentences on the path. Any facts not so specified by the minimal sentences may be filled in arbitrarily.

Note, incidentally, that this recipe gives a consistent interpretation. Since the path is open, it cannot contain both an atomic sentence and its negation. So this recipe will not make an atomic sentence both true and false. That is, it will not both say and deny that a predicate is true of an object.

The main work we need to do is to prove the analogy of lemma 18, namely

> L37: The sentences of the initial tableau are all true in a terminal interpretation represented by an open path.

We prove this by proving that a terminal interpretation makes true all the sentences in all the tableaux along its path, arguing by induction on the length of the sentences.

We need to take a little care in saying what the length of a sentence is. To keep things initially simple, let us first consider a special case—analogous to our procedure in section 13–4: Suppose that 'v', '~', and the quantifiers are the only connectives occurring in any of the initial sentences. Then we can take the length of a sentence simply to be the number of connectives occurring in the sentence.

To carry out the inductive argument, suppose that we have an open path and a terminal interpretation, **I**, represented by that path. By the definition of a terminal interpretation, all atomic and negated atomic sentences, and so all sentences of length 0 or 1, along this path are true in **I**.

For the inductive hypothesis, suppose that all sentences of length no greater than n along the path are true in **I**. Let **X** be a sentence of length n + 1. Suppose that **X** has the form ~(**Y**v**Z**). Then rule R2 for sequential generation tells us that ~**Y** and ~**Z** will both be on the path, since they will be on the tableau generated by the tableau on which ~(**Y**v**Z**) occurs. ~**Y** and ~**Z** are both shorter than ~(**Y**v**Z**), and the inductive hypothesis tells us that ~**Y** and ~**Z** are both true in **I**. Hence ~(**Y**v**Z**), that is, **X**, is true in **I**. When **X** has the form ~~**Y** the argument goes quite like the case of ~(**Y**v**Z**).

Next, we must consider **X** of the form **Y**v**W**. Such **X** gives rise to two generated tableaux, one including **Y** and one including **W**. One of these generated tableaux must be on the open path. Suppose it is the one with **Y**. Since (by the inductive hypothesis) all sentences along this path with n or fewer connectives are true, **Y**, and so **Y**v**W**, are true. If **W** rather than **Y** is on the path, the same argument applies.

Suppose that **X** has the form (∃**u**)**P**(**u**). Then rule R3′ specifies that there is a subderivation along the path that includes a substitution instance, **P**(**s**), which the inductive hypothesis tells us is true in **I**. Definition D22 applies to tell us that then (∃**u**)**P**(**u**), that is, **X**, is true in **I**.

Now suppose that **X** has the form (∀**u**)**P**(**u**). Rule R2′ specifies that, for each tableau in which (∀**u**)**P**(**u**) appears, all its substitution instances formed with names in that tableau appear in the next sequentially generated tableau. (∀**u**)**P**(**u**) is also reiterated, so that any name which comes up will eventually get instantiated along the path. By the inductive hypothesis, all these substitution instances are true in **I**. Remember that in a terminal interpretation there is exactly one object named by each name, and we have just seen that all of these names eventually get used to form true substitution instances of (∀**u**)**P**(**u**). So lemma L32 applies to tell us that (∀**u**)**P**(**u**), that is, **X**, is also true in **I**.

Make sure that you understand how this last step in the inductive proof makes essential use of the fact that (∀**u**)**P**(**u**) is always reiterated, to ensure that when new names come up in later tableaux, they will always be used to instantiate (∀**u**)**P**(**u**).

We still need to consider sentences of the form ~(∃**u**)**P**(**u**) and ~(∀**u**)**P**(**u**). Rule R2′ applies to such sentences to produce sentences, respectively, of the form (∀**u**)~**P**(**u**) and (∃**u**)~**P**(**u**). There might seem to be a problem here because ~(∃**u**)**P**(**u**) and (∀**u**)~**P**(**u**) have the same number of connectives, as do ~(∀**u**)**P**(**u**) and (∃**u**)~**P**(**u**). But we can still complete the inductive step. Suppose that ~(∃**u**)**P**(**u**) has n + 1 connectives and appears on the path. R2′ tells us that (∀**u**)~**P**(**u**), also having n + 1 connectives, also appears on the path. But we have already seen that the inductive hypothesis ensures us of the truth of (∀**u**)~**P**(**u**) in the terminal interpretation, **I**. Lemma L29 then tells us that ~(∃**u**)**P**(**u**) is also true in **I**. Of course, the case for ~(∀**u**)**P**(**u**) works the same way.

To complete the proof of L37 we must lift the restriction and allow

sentences to include all the sentence logic connectives. This creates a new difficulty. For example, R2 instructs us to generate (**X**&**Y**)v(~**X**&~**Y**) from **X**≡**Y**. But (**X**&**Y**)v(~**X**&~**Y**) has four **more** connectives than **X**≡**Y** rather than fewer.

We can resolve this impasse by assigning weights to the connectives. '~', 'v', and the quantifiers are each worth one "point," '⊃' and '&' each get three "points," and '≡' gets six "points." The length of a sentence is now just the number of these "points" added up for all the connectives in the sentence. (This technique can also be applied to arrange for ~(∃**u**)**P**(**u**) and ~(∀**u**)**P**(**u**) to be longer than (∀**u**)~**P**(**u**) and (∃**u**)~**P**(**u**).)

EXERCISE

15–23. Complete the inductive step of the argument for lemma L37 with all of the sentence logic connectives.

We have proved L37, the analogue of L18 needed for proving completeness for sentence logic derivations. The proof for sentence logic derivations also used L20, which says that if all terminal tableaux close, then 'A&~A' appears as the derivation's final conclusion. We must reformulate the statement ever so slightly because, with the possibility of infinite derivations, some paths might not have terminal tableaux. So we will say

D25: a semantic tableau derivation is *Closed* if all sequences of subderivations terminate in a closed tableau.

You will then prove the analogy of L20:

L38: If a semantic tableau derivation is closed, then 'A&~A' appears as the derivation's final conclusion.

The key to L20 is the inductive step, L21. Again, we only need to reformulate to accommodate our more specific definition of a closed tableau derivation:

L39: Let **D** be a closed semantic tableau derivation. Then, if all of **D**'s subderivations of level i have 'A&~A' as their final conclusion, so do all the subderivations of level i + 1.

EXERCISE

15–24. Prove L39. You only need to check the inductive step for rule R3′, involving subderivations started with a substitution instance of a sentence of the form (∃**u**)**P**(**u**). Be sure you see how the new

name requirement in the statement of R3′ functions crucially in your proof.

If you now go back and read the short paragraph proving T7 and change just the words 'L18' and 'L20' to 'L37' and 'L38', you will see that we have a proof of T7, where the set of sentences **Z** may now include predicate logic sentences. T7 applies exactly as it did in section 12-3 to establish

T15 (Completeness for predicate logic derivations): For any finite set of sentences, **Z**, and any sentence **X**, if $\mathbf{Z} \models \mathbf{X}$, then $\mathbf{Z} \vdash \mathbf{X}$.

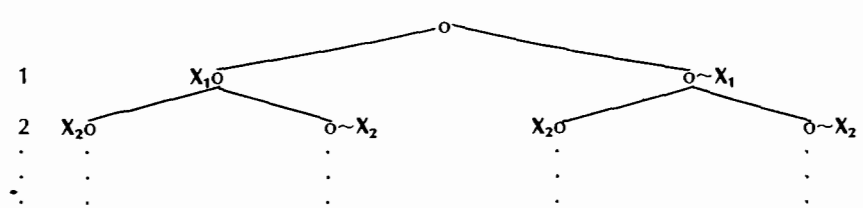
15–5. COMPACTNESS, IDENTITY, AND FUNCTIONS

In this section I am going to get started in cleaning up some details. But I am going to let you do most of the work. Students of truth trees and of derivations will be able to apply the material of this section appropriately to what they have learned.

My completeness proofs for predicate logic assumed a finite set of sentences, **Z**. To get a full statement of completeness, where **Z** can be infinite, we need to show that the compactness result, T8, which we proved in chapter 14, also holds for predicate logic. To accomplish this we need to modify the idea of a tree of truth value assignments.

Here's what we do. We can consider all possible closed atomic sentences written out in some definite order: the first atomic sentence letter, the second, the first one place predicate with the first name, the second . . . : 'A', 'B', 'Pa', 'Pb', 'Raa'. . . . To make sure that this is possible, again consider that we could write each such description of the atomic sentences in English and order them as in a dictionary.

Say the closed atomic sentences are $\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, . . . Then we can diagram all possible truth value assignments to these atomic sentences in the form of a tree:



The third line will catalogue the alternative truth values for $\mathbf{X_3}$ underneath all the possibilities covered in lines 1 and 2, and so on.

Note that each path through this tree represents an interpretation, indeed, just the sort of interpretations represented by open paths on a truth tree or semantic tableau derivation. We have seen, in the completeness proofs, how there must be at least one such interpretation for each consistent finite set of sentences. We now proceed very much as we did in predicate logic. Let **Z** be an infinite set of sentences all of the finite subsets of which are consistent. We list the sentences in some definite order, and consider the initial finite segments of this ordering: $\mathbf{Z_1}$, $\mathbf{Z_2}$, $\mathbf{Z_3}$, . . . As we work down the lines of the tree, we close branches which conflict with some sentences in one of the $\mathbf{Z_i}$. Since all of the $\mathbf{Z_i}$ are consistent, for each line of the tree, there will be an open branch reaching down to that line. Koenig's lemma tells us that there is then an infinite path through the tree. But (if you make the right sort of arrangement of when paths get closed) you will see that this infinite path represents an interpretation which makes true all the sentences in all the $\mathbf{Z_i}$. That is, this interpretation is a model of **Z**.

EXERCISE

15–25. Following the suggestions of the argument sketch given in the last paragraph, give a detailed proof of compactness for predicate logic.

Actually, we have done the work to prove the *Löwenheim Skolem Theorem*, a much stronger result, of fundamental importance in logic and set theory. In all my discussion of infinite interpretations, I have not mentioned the fact that there are different kinds of infinities. The infinity of the integers is the smallest, called, for obvious reasons, a *Countable Infinity*. However, other infinities are, in a certain sense, "larger." Consider, for example, the infinity of the real numbers (numbers representable by a finite or an infinite decimal fraction, such as 27.75283 . . .). The infinity of the real numbers is larger, or *Uncountable*, in the sense that there is no one-to-one correspondence between the integers and the real numbers. We cannot list the real numbers with the integers the way we can an infinite set of sentences.

The Löwenheim Skolem theorem says that if a set of sentences has a model with a finite, countable, or uncountable domain, then it has a finite or a countable model. For finite sets of sentences, these models are generated by open paths on a truth tree or semantic tableau derivation. If a finite set has a model (finite, countable, or uncountable) then there is an open path. But then the open path represents a finite or countably infinite model. The compactness theorem then shows how the same is true of infinite consistent sets of sentences. (If our object language does not

include identity, then there is always a countable model. But '=' allows us to write a sentence which, for example, is only true in an interpretation with exactly one object. Can you produce such a sentence?)

My soundness and consistency proofs assumed that our object language contained neither identity nor function symbols. For the moment, let's consider just identity. To begin with, we must refine the characterization of an interpretation with requirements which should seem natural if '=' really means 'identity':

> D20′ (Interpretations for languages with identity): An interpretation is as described in D20 with the following two additional requirements:
>
> a) A sentence of the form **s=t** is true in an interpretation iff **s** and **t** name the same object.
> b) For all **atomic** sentences of the form **R(s,t)**, if **s=t** is true in an interpretation, then **R(s,t)** and **R′(s,t)** have the same truth value in the interpretation, where **R′(s,t)** arises from **R(s,t)** by replacing any number of occurrences of **s** with **t** or of **t** with **s**.

Clause b) covers sentences such as 'Qab': If 'a=c' is true in an interpretation, then 'Qab' and 'Qcb' have the same truth value in the interpretation.

A good many of the semantical facts surrounding identity turn on the following lemma, which simply generalizes clause b) to the case of any closed sentence:

> L40: Let **I** be an interpretation for predicate logic with identity. Then, for all sentences of the form **R(s,t)**, if **s=t** is true in **I**, **R(s,t)** and **R′(s,t)** have the same truth value in **I**, where **R′(s,t)** arises from **R(s,t)** by replacing any number of occurrences of **s** with **t** or of **t** with **s**.

EXERCISE

15–26. Prove L40.

You are now in a position to examine how our soundness proofs need to be modified if our language includes identity. Identity involves new rules, the roles of which need to be checked in the proofs.

EXERCISES

15–27. (Trees) Show that the truth tree = rule is downwardly correct. To treat the ≠ rule, note that we can reconstrue it in the following way: Whenever a sentence of the form **s≠s** appears on a branch, also write the sentence **s=s** on that branch. Explain why this rule comes to the same as the ≠ rule as stated in chapter 9. Prove that the rule in this form is downwardly correct.

15–28. (Derivations) State and prove rule soundness for the two derivation rules for identity. Comment on whether and, if so, how these rules require any changes in the inductive proof of soundness for derivations.

We can turn now to completeness. For semantic tableau derivations we must add two new parts to the rules for sequential generation, corresponding exactly to the =I and =E rules: Whenever a name **s** occurs on a tableau, include the sentence **s=s** on the sequentially generated tableau. And if two sentences of the form **s=t** and **R(s,t)** appear on a tableau, include the sentences **R′(s,t)** on the sequentially generated tableau. Then, for both trees and semantic tableau derivations, we change how we read an interpretation off an open branch. Before, every name was assigned a distinct object. Now each name will be assigned a distinct object unless a sentence of the form **s=t** appears on the branch. Then **s** and **t** are assigned the same object. This corresponds to clause a) in D20′. Clause b) in D20′ is already ensured by the identity rules for trees and for tableau generation.

EXERCISES

15–29. (Trees) Show that clause b) of D20′ will be satisfied in the interpretation represented by an open branch. Comment on the status of lemma L40 in describing an open branch. That is, note the way in which, in effect, proof of upward adequacy automatically covers the work done by lemma L40. Then check that the tree method with identity is upwardly adequate. Though intuitively quite clear, a formal proof requires care, since the input and output sentences for the = rule all have the same predicates and connectives, so that none of our prior methods of attributing lengths to sentences will apply here.

15–30. (Derivations) Show that clause b) of D20′ will be satisfied in the interpretation represented by an open branch. Comment on the status of lemma L40 in describing an open branch. That is, note the way in which, in effect, proof of lemma L37 automatically covers the work done by lemma L40. Then check that lemma L37 is still correct. Just as with the case for trees, proof requires care, since none of our prior means of assigning lengths to sentences will work here.

Finally, let's take a brief look at function symbols. Again, we must extend the definition of an interpretation:

> D20″ (Interpretations for languages with function symbols): An interpretation is as described in D20 or D20′, with the following addition: For each function symbol, **f**, and each object, **o**, in the domain of the interpretation, the interpretation assigns a unique object **o′** = **f(o)**, as the value of **f** applied to **o**. If **s** is a closed term referring to object **o***, then **f(s)** is a term referring to **f(o*)**.

The last sentence in D20″ constitutes a recursive definition. If **s** is a name, referring to **o,** then **f(s)** refers to **f(o)**, **ff(s)** refers to **ff(o)**, and so on.

As with identity, once we have made this extension of the notion of an interpretation, most of the work is done.

EXERCISES

15–31. (Trees) Check the downward correctness of the quantifier rules when the language includes function symbols.

15–32. (Derivations) Check the proof of rule soundness for the quantifier rules when the language includes function symbols.

15–33. (Trees) Check that the proof of upward adequacy works when interpretations are read off open branches in accord with definition D20″.

15–34. (Derivations) Check lemma L37 when interpretations are read off open branches in accord with definition D20″.

15–6. CONCLUSION

You have worked hard trying to understand these proofs of soundness and completeness. I too have worked hard, first in understanding them and then in my efforts to write them up in a clear and accessible form. Working on the strength of the presentations of others, I will be very happy if I have made some small contribution to improving the accessibility of soundness and completeness and if I have avoided both horns of the dilemma of too much complication versus inaccuracies in the proofs. Whatever I have accomplished, I am sure that my presentation can be improved. I welcome your comments and suggestions. In the meantime, you should not be discouraged if you have found part II of this text to be very difficult. Soundness and completeness are substantial mathematical results. If you understand them only in a fragmentary way, you can greatly improve your grasp by patiently going over these chapters again.

CHAPTER CONCEPTS

In reviewing this chapter, be sure you have a firm grasp on the following:

a) Interpretation
b) Unnamed Object
c) **s**-Variant
d) Truth of an Existentially Quantified Sentence
e) Truth of a Universally Quantified Sentence
f) Infinite Tree
g) Infinite Semantic Tableau Derivation
h) Downward correctness of a Truth Tree Rule
i) Upward correctness of a Truth Tree Rule
j) Interpretation Represented by an Infinite Truth Tree Branch
k) Restrictions on the Derivation Rule for ∀I
l) Restrictions on the Derivation Rule for ∃E
m) Terminal Interpretation in a Semantic Tableau Derivation
n) Closed Semantic Tableau Derivation
o) Compactness for Predicate Logic
p) Interpretation for a Language with Identity
q) Interpretation for a Language with Function Symbols