Identity, Functions, and Definite Descriptions

9

9–1. IDENTITY

Clark Kent and Superman would seem to be entirely different people. Yet it turns out they are one and the same. We say that they are *Identical.* Since identity plays a special role in logic, we give it a permanent relation symbol. We express 'a is identical to b' with 'a=b', and the negation with either '~(a=b)' or 'a≠b'.

'=' is not a connective, which forms longer sentences from shorter sentences. '=' is a new logical symbol which we use to form atomic sentences out of names and variables. But as we did with the connectives, we can explain exactly how to understand '=' by giving truth conditions for closed sentences in interpretations. Just follow the intuitive meaning of identity: To say that **s**=**t** is to say that the thing named by **s** is identical to the thing named by **t**; that is, that the names **s** and **t** refer to the same object. (Logicians say that **s** and **t** have the same referent, or that they are *Co-Referential.*) To summarize

> '=' flanked by a name or a variable on either side is an atomic sentence. If **s** and **t** are names, **t**=**s** is true in an interpretation if **s** and **t** name the same thing. **s**=**t** is false if **s** and **t** name different things. The negation of an identity sentence can be written either as ~(**s**=**t**) or as **s**≠**t**.

Identity is easy to understand, and it is extraordinarily useful in expressing things we could not say before. For example, '(∃x)' means that

there is one **or more** x such that. . . . Let's try to say that there is exactly one x such that . . . , for which we will introduce the traditional expression '(∃x!)' (read "E shriek"). We could, of course, introduce '(∃x!)' as a new connective, saying, for example, that '(∃x!)Bx' is true in an interpretation just in case exactly one thing in the interpretation is B. But, with the help of identity, we can get the same effect with the tools at hand, giving a rewriting rule for '(∃x!)' much as we did for subscripted quantifiers in chapter 4.

To say that there is exactly one person (or thing) who is blond is to say, first of all, that someone is blond. But it is further to say that nothing else is blond, which we can reexpress by saying that if anything is blond, it must **be** (that is, be identical to) that first blond thing. In symbols, this is '(∃x)[Bx & (∀y)(By ⊃ y=x)]'.

Before giving a general statement, I want to introduce a small, new expository device. Previously I have used the expression '(. . . **u** . . .)' to stand for an arbitrary sentence with **u** the only free variable. From now on I am going to use expressions such as **P(u)** and **Q(u)** for the same thing:

> Boldface capital letters followed by a variable in parentheses, such as **P(u)** and **Q(u)**, stand for arbitrary sentences in which **u**, and only **u**, may be free. Similarly, **R(u,v)** stands for an arbitrary sentence in which at most **u** and **v** are free.

In practice **P(u)**, **Q(u)**, and **R(u,v)** stand for open sentences with the indicated variable or variables the only free variable. However, for work in Part II of this Volume, I have written the definition to accommodate degenerate cases in which **u**, or **u** and **v**, don't actually occur or don't occur free. If you are not a stickler for detail, don't worry about this complication: Just think of **P(u)**, **Q(u)**, and **R(u,v)** as arbitrary open sentences. But if you want to know why I need, to be strictly correct, to cover degenerate cases, you can get an idea from exercise 13–3.

With this notation we can give the E! rewrite rule:

> *Rule for rewriting* ∃!: For any open formula **P(u)** with **u** a free variable, (∃**u**!)**P(u)** is shorthand for (∃**u**)[**P(u)** & (**v**)(**P(v)** ⊃ **v**=**u**)], where **v** is free for **u** in **P(u)**, that is, where **v** is free at all the places where **u** is free in **P(u)**.

Once you understand how we have used '=' to express the idea that exactly one of something exists, you will be able to see how to use '=' to express many related ideas. Think through the following exemplars until you see why the predicate logic sentences expresses what the English expresses:

> There are at least two x such that Fx:
> (∃x)(∃y)[x≠y & Fx & Fy].

There are exactly two x such that Fx:
$(\exists x)(\exists y)\{x \neq y \,\&\, Fx \,\&\, Fy \,\&\, (\forall z)[Fz \supset (z = x \vee z = y)]\}$.

There are at most two x such that Fx:
$(\forall x)(\forall y)(\forall z)[(Fx \,\&\, Fy \,\&\, Fz) \supset (x = y \vee x = z \vee y = z)]$.

We can also use '=' to say some things more accurately which previously we could not say quite correctly in predicate logic. For example, when we say that everyone loves Adam, we usually intend to say that everyone **other** than Adam loves Adam, leaving it open whether Adam loves himself. But '$(\forall x)$' means absolutely everyone (and thing), and thus won't exempt Adam. Now we can use '=' explicitly to exempt Adam:

Everyone loves Adam (meaning, everyone except possibly Adam himself):
$(\forall x)(x \neq a \supset Lxa)$.

In a similar way we can solve a problem with transcribing 'Adam is the tallest one in the class'. The problem is that no one is taller than themself, so we can't just use '$(\forall x)$', which means absolutely everyone. We have to say explicitly that Adam is taller than all class members **except Adam**.

Adam is the tallest one in the class:
$(\forall x)[(Cx \,\&\, x \neq a) \supset Tax]$.

To become familiar with what work '=' can do for us in transcribing, make sure you understand the following further examples:

Everyone except Adam loves Eve:
$(\forall x)(x \neq a \supset Lxe) \,\&\, \sim Lae$.

Only Adam loves Eve:
$(\forall x)(Lxe \equiv x = a)$, or $Lae \,\&\, (\forall x)(Lxe \supset x = a)$.

Cid is Eve's only son:
$(\forall x)(Sxe \equiv x = c)$, or $Sce \,\&\, (\forall x)(Sxe \supset x = c)$.

EXERCISES

9–1. Using Cx: x is a clown, transcribe the following:

a) There is at least one clown.
b) There is no more than one clown.
c) There are at least three clowns.
d) There are exactly three clowns.
e) There are at most three clowns.

9–2. Use the following transcription guide:

a:	Adam	Sxy:	x is smarter than y
e:	Eve	Qxy:	x is a parent of y
Px:	x is a person	Oxy:	x owns y
Rx:	x is in the classroom	Mxy:	x is a mother of y
Cx:	x is a Cat		
Fx:	x is furry		

Transcribe the following:

a) Three people love Adam. (Three or more)
b) Three people love Adam. (Exactly three)
c) Eve is the only person in the classroom.
d) Everyone except Adam is in the classroom.
e) Only Eve is smarter than Adam.
f) Anyone in the classroom is smarter than Adam.
g) Eve is the smartest person in the classroom.
h) Everyone except Adam is smarter than Eve.
i) Adam's only cat is furry.
j) Everyone has exactly one maternal grandparent.
k) No one has more than two parents.

9–2. INFERENCE RULES FOR IDENTITY

You now know what '=' means, and you have practiced using '=' to say various things. You still need to learn how to use '=' in proofs. In this section I will give the rules for '=' both for derivations and for trees. If you have studied only one of these methods of proof, just ignore the rules for the one you didn't study.

As always, we must guide ourselves with the requirement that our rules be truth preserving, that is, that when applied to sentences true in an interpretation they should take us to new sentences also true in that interpretation. And the rules need to be strong enough to cover all valid arguments.

To understand the rules for both derivations and trees, you need to appreciate two general facts about identity. The first is that everything is self-identical. In any interpretation which uses the name 'a', 'a = a' will be true. Thus we can freely use statements of self-identity. In particular, self-identity should always come out as a logical truth.

The second fact about identity which our rules need to reflect is just this: If a = b, then anything true about a is true about b, and conversely.

I'm going to digress to discuss a worry about how general this second fact really is. For example, if Adam believes that Clark Kent is a weakling and if in addition Clark Kent is Superman, does it follow that Adam believes that Superman is a weakling? In at least one way of understanding these sentences the answer must be "no," since Adam may well be laboring under the false belief that Clark Kent and Superman are different people.

Adam's believing that Clark Kent is a weakling constitutes an attitude on Adam's part, not just toward a person however named, but toward a person known under a name (and possibly under a further description as well). At least this is so on one way of understanding the word 'believe'. On this way of understanding 'believe', Adam's attitude is an attitude not just about Clark Kent but about Clark Kent under the name 'Clark Kent'. Change the name and we may change what this attitude is about. What is believed about something under the name 'a' may be different from what is believed about that thing under the name 'b', whether or not in fact a = b.

This problem, known as the problem of substitutivity into belief, and other so-called "opaque" or "intensional" contexts, provides a major research topic in the philosophy of language. I mention it here only to make clear that predicate logic puts it aside. An identity statement, 'a = b', is true in an interpretation just in case 'a' and 'b' are names of the same thing in the interpretation. Other truths in an interpretation are specified by saying which objects have which properties, which objects stand in which relations to each other, and so on, irrespective of how the objects are named. In predicate logic all such facts must respect identity.

Thus, in giving an interpretation of a sentence which uses the predicate 'B', one must specify the things in the interpretation, the names of these things, and then the things of which 'B' is true and the things of which 'B' is false. It is most important that this last step is independent of which names apply to which objects. Given an object in the interpretation's domain, we say whether or not 'B' is true of that **object**, however that thing happens to be named. Of course, we may use a name in saying whether or not 'B' is true of an object—indeed, this is the way I have been writing down interpretations. But since interpretations are really characterized by saying which predicates apply to which objects, if we use names in listing such facts, we must treat names which refer to the same thing, so-called *Co-Referential Names*, in the same way. If 'a' and 'b' are names of the same thing and if we say that 'B' is true of this thing by saying that 'Ba' is true, then we must also make 'Bb' true in the interpretation.

In short, given the way we have defined truth in an interpretation, if 'a = b' is true in an interpretation, and if something is true of 'a' in the interpretation, then the same thing is true of 'b' in the interpretation. Logicians say that interpretations provide an *Extensional Semantics* for predicate logic. "Semantics" refers to facts concerning what is true, and facts concerning meaning, insofar as rules of meaning have to do with what comes out true in one or another circumstance. "Extensional" means that the *Extension* of a predicate—the collection of things of which the predicate is true—is independent of what those things are called. Parts of English (e.g., 'Adam believes Clark Kent is a weakling') are not extensional. Predicate logic deals with the special case of extensional sentences. Because predicate logic deals with the restricted and special case of extensional sentences, in predicate logic we can freely substitute one name for another when the names name the same thing.

Now let's apply these two facts to write down introduction and elimination rules for identity in derivations. Since, for any name, **s**, **s = s** is always true in an interpretation, at any place in a derivation which we can simply introduce the identity statement **s = s**:

| s = s   =I   Where **s** is any name.

If **s** does not occur in any governing premises or assumptions, it occurs arbitrarily and gets a hat. To illustrate, let's demonstrate that '(∀x)(x = x)' is a logical truth:

```
1 | â = â          =I
2 | (∀x)(x = x)    2, ∀I
```

The second fact, that co-referential names can be substituted for each other, results in the following two rules:

```
| s = t            | s = t
| P(s)             | P(t)
|                  |
| P(t)   =E        | P(s)   =E
```

The indicated substitutions may be for any number of occurrences of the name substituted for.

To illustrate, let's show that '(∀x)(∀y)[x = y ⊃ (Fx ⊃ Fy)]' is a logical truth:

```
1 | | a=b                              A
  | |----
2 | | | Fa                             A
  | | |----
3 | | | a=b                            1, R
4 | | | Fb                             2, 3, =E
5 | | Fa ⊃ Fb                          2–4, ⊃I
6 | âb̂ ⊃ (Fâ ⊃ Fb̂)                    1–5, ⊃I
7 | (∀y)[â=y ⊃ (Fâ ⊃ Fy)]            6, ∀I
8 | (∀x)(∀y)[x=y ⊃ (Fx ⊃ Fy)]        7, ∀I
```

Now we'll do the rules for trees. We could proceed much as we did with derivations and require that we write identities such as 'a=a' wherever this will make a branch close. An equivalent but slightly simpler rule instructs us to close any branch on which there appears a negated self-identity, such as 'a≠a'. This rule makes sense because a negated self-identity is a contradiction, and if a contradiction appears on a branch, the branch cannot represent an interpretation in which all its sentences are true. In an exercise you will show that this rule has the same effect as writing self-identities, such as 'a=a', wherever this will make a branch close.

> *Rule* ≠: For any name, **s**, if **s**≠**s** appears on a branch, close the branch.

Let's illustrate by proving '(∀x)(x=x)' to be a logical truth:

```
√1  ~(∀x)(x=x)     ~S
√2   (∃x)(x≠x)      1, ~∀
 3     a≠a          2, ∃
        ×
```

The second rule for trees looks just like the corresponding rules for derivations. Substitute co-referential names:

> *Rule* =: For any names, **s** and **t**, if **s**=**t** appears on a branch, substitute **s** for **t** and **t** for **s** in any expression on the branch, and write the result at the bottom of the branch if that sentence does not already appear on the branch. Cite the line numbers of the equality and the sentence into which you have substituted. But do not check either sentence. Application of this rule to a branch is not completed until either the branch closes or until all such substitutions have been made.

Let's illustrate, again by showing '(∀x)(∀y)[x=y ⊃ (Fx ⊃ Fy)]' to be a logical truth:


```
√1  ~(∀x)(∀y)[x=y ⊃ (Fx ⊃ Fy)]   ~S
√2  (∃x)(∃y)~[x=y ⊃ (Fx ⊃ Fy)]   1, ~∀, ~∀
√3      ~[a=b ⊃ (Fa ⊃ Fb)]       2, ∃, ∃
 4           a=b                 3, ~⊃
√5        ~(Fa ⊃ Fb)             3, ~⊃
 6            Fa                 5, ~⊃
 7           ~Fb                 5, ~⊃
 8           ~Fa                 4, 7, =
              ×
```

Before closing this discussion of identity, I should mention that identity provides an extreme example of what is called an *Equivalence Relation*. Saying that identity is an equivalence relation is to attribute to it the following three characteristics:

> Identity is *Reflexive*. Everything is identical with itself: (∀x)(x=x). In general, to say that relation **R** is reflexive is to say that (∀x)R(x,x).
>
> Identity is *Symmetric*. If a first thing is identical with a second, the second is identical with the first: (∀x)(∀y)(x=y ⊃ y=x). In general, to say that relation **R** is symmetric is to say that (∀x)(∀y)(**R**(x,y) ⊃ **R**(y,x)).
>
> Identity is *Transitive*. If a first thing is identical with a second, and the second is identical with a third, then the first is identical with the third: (∀x)(∀y)(∀z)[(x=y & y=z) ⊃ x=z]. In general, to say that relation **R** is transitive is to say that (∀x)(∀y)(∀z)[(**R**(x,y) & **R**(y,z)) ⊃ **R**(x,z)].

You can prove that identity is an equivalence relation using either derivations or trees.

Here are some other examples of equivalence relations: being a member of the same family, having (exactly) the same eye color, being teammates on a soccer team. Items which are related by an equivalence relation can be treated as the same for certain purposes, depending on the relation. For example, when it comes to color coordination, two items with exactly the same color can be treated interchangeably. Identity is the extreme case of an equivalence relation because "two" things related by identity can be treated as the same for **all** purposes.

Equivalence relations are extremely important in mathematics. For example two numbers are said to be *Equal Modulo 9* if they differ by an exact multiple of 9. Equality modulo 9 is an equivalence relation which is useful in checking your arithmetic (as you know if you have heard of the "rule of casting out 9s").

EXERCISES

9–3. Show that each of the two =E rules can be obtained from the other, with the help of the =I rule.

9–4. Show that the rule $\neq$ is equivalent to requiring one to write, on each branch, self-identities for each name that occurs on the branch.

Do the following three exercises using derivations, trees, or both:

9–5. Show that the following are logical truths:

a) $(\exists x)(x=a)$
b) $(\forall x)(\forall y)[\sim(Fx \supset Fy) \supset x \neq y]$
c) $(\forall x)[Px \equiv (\exists y)(x=y \,\&\, Py)]$
d) $Pa \equiv (\forall x)(x=a \supset Pa)$
e) $(\exists x)(\exists y)(Fx \,\&\, \sim Fy) \supset (\exists x)(\exists y)(x \neq y)$

9–6. Show that $(\exists x)(\forall y)(Fy \equiv y=x)$ and $(\exists x!)Fx$ are logically equivalent.

9–7. Prove that $=$ is an equivalence relation.

9–8. Show the validity of the following arguments:

a) $\begin{array}{c} (\forall x)(x=a \supset Fx) \\ \hline Fa \end{array}$ b) $\begin{array}{c} Fa \\ \hline (\forall x)(x=a \supset Fx) \end{array}$ c) $\begin{array}{c} (\exists x)(Fx \,\&\, x=a) \\ \hline Fa \end{array}$

d) $\begin{array}{c} (\forall x)(x=a \supset Fx) \\ (\forall x)(Fx \supset Fb) \\ \hline Fb \end{array}$ e) $\begin{array}{c} Pa \\ (\exists y)(y=a \,\&\, y=b) \\ \hline Pb \end{array}$ f) $\begin{array}{c} a=b \\ \hline Fa \equiv Fb \end{array}$

g) $\begin{array}{c} a=b \\ \hline (\forall x)(a=x \equiv b=x) \end{array}$ h) $\begin{array}{c} (\forall x)(a=x \equiv b=x) \\ \hline a=b \end{array}$ i) $\begin{array}{c} (\exists x)(\forall y)(Py \equiv y=x) \\ Pa \\ Pb \\ \hline a=b \end{array}$

j) $\begin{array}{c} (\exists x)Px \\ (\forall x)(x=a \vee x=b) \\ \hline Pa \vee Pb \end{array}$ k) $\begin{array}{c} (\exists x)(\forall y)(x=y) \\ \hline (\exists x)Px \supset (\forall x)Px \end{array}$ l) $\begin{array}{c} (\forall x)(\exists y)Rxy \\ (\forall x)\sim Rxx \\ \hline (\exists x)(\exists y)(x \neq y) \end{array}$

m) $\begin{array}{c} (\forall x)(\exists y)Rxy \\ (\forall x)\sim Rxx \\ \hline (\forall x)(\exists y)(Rxy \,\&\, x \neq y) \end{array}$ n) $\begin{array}{c} (\exists x)(Kx \,\&\, Jx) \\ (\exists x)(Kx \,\&\, \sim Jx) \\ \hline (\exists x)(\exists y)(Kx \,\&\, Ky \,\&\, x \neq y) \end{array}$

o) $\begin{array}{c} (\exists x!)Px \\ (\exists x)(Px \,\&\, Qx) \\ \hline (\forall x)(Px \supset Qx) \end{array}$ p) $\begin{array}{c} \sim(\exists !x)Fx \\ (\exists x)Fx \\ \hline (\exists x)(\exists y)(Fx \,\&\, Fy \,\&\, x \neq y) \end{array}$

9–9. I stated that being teammates on a soccer team is an equivalence relation. This is right, on the assumption that no one belongs to more than one soccer team. Why can the relation, *being teammates on a soccer team,* fail to be an equivalence relation if someone belongs to two teams? Are there any circumstances under which being teammates on a soccer team is an equivalence relation even though one or more people belong to more than one team?

9–3. FUNCTIONS

Often formal presentations of functions leave students bewildered. But if you have done any high school algebra you have an intuitive idea of a function. So let's start with some simple examples from algebra.

For our algebraic examples, the letters 'x', 'y', and 'z' represent variables for numbers. Consider the expression '$y = 2x + 7$'. This means that if you put in the value 3 for x you get the value $2\times 3 + 7 = 13$ for y. If you put in the value 5 for x, you get the value $2\times 5 + 7 = 17$ for y. Thus the expression '$y = 2x + 7$' describes a rule or formula for calculating a value for y if you give it a value for x. The formula always gives you a definite answer. Given some definite value for x, there is exactly one value for y which the formula tells you how to calculate.

Mathematicians often use expressions like '$f(x)$' for functions. Thus, instead of using the variable y in the last example, I could have written '$f(x) = 2x + 7$' This means exactly what '$y = 2x + 7$' means. When you put in a specific number for x, '$f(x)$' serves as a name for the value y, so that we have $y = f(x)$. In particular, '$f(3)$' is a name for the number which results by putting in the value 3 for x in $2x + 7$. That is, '$f(3)$' is a name for the number 13, the number which results by putting in the value 3 for x in $f(x) = 2x + 7$.

This is all there is to functions in logic. Consider the name 'a'. Then '$f(a)$' acts like another name. To what does '$f(a)$' refer? That depends, of course, on what function $f(\;)$ is, which depends on how '$f(\;)$' is interpreted. In specifying an interpretation for a sentence in which the function symbol '$f(\;)$' occurs, we must give the rule which tells us, for any name **s**, what object $f(\mathbf{s})$ refers to. When we deal with interpretations in which there are objects with no names, this must be put a little more abstractly: We must say, for each object (called the *Argument* of the function), what new object (called the *Value* of the function) is picked out by the function $f(\;)$ when $f(\;)$ is applied to the first object. The function must be well defined, which means that for each object to which it might be applied, we must specify exactly **one** object which the function picks out. For each argument there must be a unique value.

So far I have talked only about one place functions. Consider the example of the mathematical formula '$z = 3x + 5y - 8$', which we can also write as '$z = g(x,y)$' or as '$g(x,y) = 3x + 5y - 8$'. Here $g(\;,\;)$ has two arguments. You give the function two input numbers, for example, $x = 2$ and $y = 4$, and the function gives you a single, unique output—in this case, the number $3\times2 + 5\times4 - 8 = 18$. Again, the idea carries over to logic. If '$g(\ ,\ )$' is a two place function symbol occurring in a sentence, in giving an interpretation for the sentence we must specify the unique object the function will pick out when you give it a **pair** of objects. If our interpretation has a name for each object the same requirement can be expressed in this way: For any two names, **s** and **t**, 'g(**s**,**t**)' refers to a unique object, the one picked out by the function $g(\ ,\ )$ when $g(\ ,\ )$ is applied to the arguments **s** and **t**. We can characterize functions with three, four, or any number of argument places in the same kind of way.

To summarize

> The interpretation of a one place function specifies, for each object in the interpretation's domain, what object the function picks out as its value when the function is applied to the first object as its argument. The interpretation of a two place function similarly specifies a value for each pair of arguments. Three and more place functions are interpreted similarly.

Incidentally, the value of a function does not have to differ from the argument. Depending on the function, these may be the same or they may be different. In particular, the trivial identity function defined by $(\forall x)(f(x) = x)$ is a perfectly well-defined function.

In the last sentence I applied a function symbol to a variable instead of a name. How should you understand such an application? In an interpretation, a name such as 'a' refers to some definite object. A variable symbol such as 'x' does not. Similarly, 'f(a)' refers to some definite object, but 'f(x)' does not. Nonetheless, expressions such as 'f(x)' can be very useful. The closed sentence '(∀x)Bf(x)' should be understood as saying that every value of 'f(x)' has the property named by 'B'. For example, let us understand 'Bx' as 'x is blond' and 'f(x)' as referring to the father of x. That is, for each person, x, f(x) is the father of x, so that 'f(a)' refers to Adam's father, 'f(e)' refers to Eve's father, and so on. Then '(∀x)Bf(x)' says that everyone's father is blond.

In sum, function symbols extend the kind of sentences we can write. Previously we had names, variables, predicate symbols, and connectives. Now we introduce function symbols as an extension of the category of names and variables. This involves the new category called *Terms:*

> We extend the vocabulary of predicate logic to include *Function Symbols,* written with lowercase italicized letters followed by parentheses with places for writing in one, two, or more arguments.
>
> All names and variables are *Terms.* A function symbol applied to any term or terms (a one place function symbol applied to one term, a two place function symbol applied to two terms, etc.) is again a term. Only such expressions are terms.

> In forming sentences, terms function exactly as do names and variables. One may be written after a one place predicate, two after a two place predicate, and so on.

Do not confuse function symbols (lowercase italicized letters followed by parentheses with room for writing in arguments) with such expressions as **P(u)** and **R(u,v)**. These latter expressions are really not part of predicate logic at all. They are part of English which I use to talk about arbitrary open predicate logic sentences.

Notice that these definitions allow us to apply functions to functions: If '$f(\ )$' is a one place function symbol, '$f(f$(a))' is a well-defined term. In practice, we leave out all but the innermost parentheses, writing '$f(f$(a))' as 'ff(a)' What does such multiple application of a function symbol mean? Well, if $f(x) = x^2$, then $ff(x)$ is the square of the square of x. If $x = 3$, then $ff(3) = (3^2)^2 = 9^2 = 81$. In general, you determine the referent of—that is, the object referred to by —'ff(a)' as follows: Look up the referent of 'a'. Apply the function f to that object to get the referent of 'f(a)'. Now apply f a second time to this new object. The object you get after the second application of f is the referent of 'ff(a)'.

Function symbols can be combined to form new terms in all kinds of ways. If '$f(\ )$' is a one place function symbol and '$g(\ ,\ )$' is a two place function symbol, the following are all terms: 'f(a)', 'f(y)', 'g(a,x)', 'fg(a,x)'—that is, $f[g$(a,x)], '$g[f$(a), f(b)]', and $g[f$(x), g(a,b)]'.

We need one more definition:

> A term in which no variables occur is called a *Constant* or a *Constant Term.*

Only constant terms actually refer to some specific object in an interpretation. But closed sentences which use nonconstant terms still have truth values. In applying the truth definitions for quantifiers, we form substitution instances, substituting names for variables within function symbols as well as elsewhere. Thus, in applying the definition for the truth of a universally quantified sentence in an interpretation to '(∀x)Laf(x)', we look at the substitution instances 'Laf(a)', 'Laf(b)', 'Laf(c)', and so on. We then look to see if the relation L holds between a and the object f(a), between a and the object f(b), and so on. Only if all these instances hold is '(∀x)Laf(x)' true in the interpretation.

The rules for functions simply reflect the fact that constant terms formed by applying function symbols to other constant terms have definite referents, just as names do. However, the generality of these new referring terms may be restricted. For example, the constant function f defined by $(\forall x)(f(x) = a)$ can only refer to one thing, namely, a. Thus, when it is important that nothing be assumed about a constant term we must use a name and not a function symbol applied to another constant term.

For derivations this means that we should treat constant terms all alike in applying the rules ∀E and ∃I. In applying ∃E, our isolated name must still be a name completely isolated to the subderivation to which the ∃E rule applies. (Strictly speaking, if you used an isolated function symbol applied to an isolated name, no difficulty would arise. But it's simpler just to let the isolated name requirement stand as a requirement to use an isolated name.)

In applying ∀I only names can occur arbitrarily. For example, we must never put a hat on a term such as '*f*(a)'. The hat means that the term could refer to absolutely **anything,** but often the value of a function is restricted to only part of an interpretation's domain. So we can't apply ∀I to a function symbol. However, if a name appears in no governing premise or assumption and occurs **as the argument** of a function symbol, we can apply ∀I to the name. For example, if 'a' appears in no governing premise or assumption, we could have 'B*f*(â)' as a line on a derivation, to which we could apply ∀I to get '(x)B*f*(x)'. To summarize

> In derivations, treat all constant terms alike in applying ∀E and ∃I. Apply ∀I and ∃E only to names.

Let's try this out by showing that '(∀x)(∃y)(*f*(x) = y)' is a logical truth. This sentence says that for each argument a function has a value. The way we treat functions in giving interpretations guarantees that this statement is true in all interpretations. If our rules are adequate, this fact should be certified by the rules:

```
1 | f(â)=f(â)          =I
2 | (∃y)(f(â)=y)       1, ∃I
3 | (∀x)(∃y)(f(x)=y)   2, ∀I
```

Note that this derivation works without any premise or assumption. =I allows us to introduce the identity of line 1. Since 'a' does not occur in any governing premise or assumption, it occurs arbitrarily, although the larger term '*f*(a)' does not occur arbitrarily. 'a' could refer to absolutely anything—that is, the argument to which the function is applied could be any object at all. However, the result of applying the function *f* to this arbitrary object might not be just anything. In line 2 we apply ∃I to the whole term '*f*(a)', not just to the argument 'a'. This is all right because we are **existentially**, not universally, generalizing. If *f*(â) =*f*(â), then *f*(â) is identical with **something**. Finally, in line 3, we universally generalize on the remaining arbitrarily occurring instance of 'a'.

Let's try something harder. '(∀x)(∃y)[*f*(x)=y & (∀z)(*f*(x)=z ⊃ z=y)]' says that for each argument the function *f* has a value and furthermore this value is unique. Again, the way we treat functions in giving interpre-

tations guarantees that this statement is true in all interpretations. So our rules had better enable us to show that this sentence is a logical truth:

```
1 | f(â)=f(â)                                 =I
2 |   | f(a)=b          A
  |   |----------
3 |   | f(a)=f(a)       =I
4 |   | b=f(a)          2, 3, =E
5 | f(â)=b̂ ⊃ b̂=f(â)                          2–4, ⊃I
6 | (∀z)(f(â)=z ⊃ z=f(â))                     5, ∀I
7 | f(â)=f(â) & (∀z)(f(â)=z ⊃ z=f(â))         1, 6, &I
8 | (∃y)(f(â)=y & (∀z)(f(â)=z ⊃ z=y)]         7, ∃I
9 | (∀x)(∃y)[f(x)=y & (∀z)(f(x)=z ⊃ z=y)]     8, ∀I
```

One more example will illustrate ∃E and ∀E as applied to terms using function symbols. Note carefully how in applying ∀E the constant term to use in this problem is not a name, but '*f*(a)', a function symbol applied to a name:

```
1 | (∃x)(∀y)[f(x)=g(y)]       P
  |-------------------------
2 |  a | (∀y)[f(a)=g(y)]     A
  |    |--------------------
3 |    | f(a)=gf(a)          2, ∀E
4 |    | (∃x)[f(x)=gf(x)]    3, ∃I
5 | (∃x)[f(x)=gf(x)]          1, 2–4, ∃E
```

Similar thinking goes into the rules for trees. All constant terms act as names when it comes to the rule ∀. But for the rule ∃ we want a name that could refer to anything in the interpretation—that was the reason for requiring that the name be new to the branch. So for ∃ we need a new name, which must be a name, not a function symbol, applied to another constant term:

> In trees, instantiate all universally quantified sentences with all constant terms that occur along the branch, unless the branch closes. Instantiate each existentially quantified sentence with a **new name**.

Let us illustrate the new rules with the same sentence as before, '(∀x)(∃y)[*f*(x)=y & (∀z)(*f*(x)=z ⊃ z=y)]'. As I mentioned, this sentence says that *f* has a unique value for each argument. Since the way we treat functions in giving interpretations ensures that this sentence is true in all interpretations, our rules had better make this sentence come out to be a logical truth:

√1	$\sim(\forall x)(\exists y)\{f(x)=y \;\&\; (\forall z)[f(x)=z \supset z=y]\}$		$\sim S$
√2	$(\exists x)(\forall y)\sim\{f(x)=y \;\&\; (\forall z)[f(x)=z \supset z=y]\}$		1, $\sim\exists$, $\sim\forall$
f(a) 3	$(\forall y)\sim\{f(a)=y \;\&\; (\forall z)[f(a)=z \supset z=y]\}$		2, $\exists$
√4	$\sim\{f(a)=f(a) \;\&\; (\forall z)[f(a)=z \supset z=f(a)]\}$		3, $\forall$
√5	$f(a)\neq f(a)$	$\sim(\forall z)[f(a)=z \supset z=f(a)]$	$\sim\&$
√6	×	$(\exists z)\sim[f(a)=z \supset z=f(a)]$	5, $\sim\forall$
√7		$\sim[f(a)=b \supset b=f(a)]$	6, $\exists$
8		$f(a)=b$	7, $\sim\supset$
9		$b\neq f(a)$	7, $\sim\supset$
10		$f(a)\neq f(a)$	8, 9, =
		×	

Notice that to get everything to close I used the term '*f*(a)' in substituting into line 3. Also, note that the right branch does not close at line 9. Line 9 is not, strictly speaking, the negation of line 8 since, strictly speaking, '*f*(a)=b' and 'b=*f*(a)' are different sentences.

The occurrence of functions in trees has an unpleasant feature. Suppose that a universally quantified sentence such as '(∀x)P*f*(x)' appears on a tree. This will be instantiated, at least once, say, with 'a', giving 'P*f*(a)'. But now we have a new constant, '*f*(a)', which we must put into '(∀x)P*f*(x)', giving 'P*ff*(a)'. This in turn gives us a further constant, '*ff*(a)'—and clearly we are off on an infinite chase. In general, open trees with function symbols are infinite when, as in '(∀x)P*f*(x)', a function symbol occurs as a nonconstant term inside the scope of a universal quantifier.

EXERCISES

9–10. Provide derivations and/or trees to establish that the following are logical truths:

a) $(\forall x)(\forall y)(\forall z)[(f(z)=x \;\&\; f(z)=y) \supset x=y]$
b) $(\exists x)[Ff(x) \vee \sim Ff(x)]$

9–11. Provide derivations and/or trees to establish the validity of the following arguments:

a) $\dfrac{(\forall x)Fx}{(\forall x)Fg(x)}$ b) $\dfrac{(\forall x)(\forall y)(x=y)}{(\forall x)(f(x)=x)}$ c) $\dfrac{(\forall x)(f(x)\neq x)}{(\exists x)(\exists y)(x\neq y)}$

d) $\dfrac{(\exists x)(f(x)\neq x)}{(\exists x)(\exists y)(fx\neq y)}$ e) $\dfrac{(\exists x)(\forall y)(f(y)=x)}{(\forall x)(\forall y)[f(x)=f(y)]}$

f) $\dfrac{(\forall x)(\forall y)[g(x,y)=g(y,x)]}{(\forall x)(\forall y)[Fg(x,y) \supset Fg(y,x)]}$ g) $\dfrac{(\forall x)(ff(x)=x)}{(\forall x)(\forall y)[f(x)=f(y) \supset x=y]}$

h) $\dfrac{(\exists x)(\forall y)(\forall z)(g(y,z)=x)}{(\forall x)(\forall y)(\forall z)(\forall w)[g(x,y)=g(z,w)]}$ i) $\dfrac{(\forall z)(\exists x)(\exists y)[z=g(x,y)]}{(\forall x)(\forall y)Fg(x,y) \supset (\forall x)Fx}$

j) $\dfrac{(\exists x)(\exists y)[Ff(x) \;\&\; \sim Ff(y)]}{(\exists x)(\exists y)[f(x)\neq f(y)]}$

k) $\dfrac{(\forall x)(\forall y)[x\neq y \supset g(x,y) \neq g(y,x)]}{(\forall x)(\forall y)\{x\neq y \supset g[g(x,y),g(y,x)] \neq g[g(y,x),g(x,y)]\}}$

l) $\dfrac{(\forall x)(\forall y)\{x\neq y \supset [Fg(x,y) \equiv \sim Fg(y,x)]\}}{(\forall x)(\forall y)[(x\neq y \supset g(x,y)\neq g(y,x)]}$

9–4. DEFINITE DESCRIPTIONS

Let's transcribe

(1) The one who loves Eve is blond.

We need a predicate logic sentence which is true when (1) is true and false when it is false. If there is exactly one person who loves Eve and this person is blond, (1) is true. If this person is not blond, (1) clearly is false. But what should we say about (1) if no one loves Eve, or more than one do?

If no one, or more than one love Eve, we surely can't count (1) as true. If we insist that every sentence is true or false, and since (1) can't be true if none or more than one love Eve, we will have to count (1) as false under these conditions. Thinking about (1) in this way results in transcribing it as

(1a) (∃x!)(Lxe & Bx).

which is true if exactly one person loves Eve and is blond, and is false if such a person exists and is not blond **or** if there are none or more than one who love Eve.

From a perspective wider than predicate logic with identity we do not have to take this stand. We could, instead, suggest that there being exactly one person who loves Eve provides a precondition for, or a *Presupposition* of, the claim that the one who loves Eve is blond. This means that the condition that there is exactly one person who loves Eve must hold for (1) to be either true or false. If the presupposition holds—if there is exactly one person who loves Eve—then (1) is true if this unique person is blond and false if he or she is not blond. If the presupposition fails—if there is none or more than one who love Eve—then we say that (1) is neither true nor false. One can design more complex systems of logic in which to formalize this idea, but predicate logic with identity does not have these resources. Hence, (1a) is the best transcription we can provide.

Grammatically, 'the one who loves Eve' functions as a term. It is supposed to refer to something, and we use the expression in a sentence by attributing some property or relation to the thing purportedly referred to. We can mirror this idea in predicate logic by introducing a new kind of expression, (The **u**)**P**(**u**), which, when there is a unique **u** which is **P**, refers to that object. We would then like to use (The **u**)**P**(**u**) like a name or other constant term in combination with predicates. Thus we would transcribe (1) as

(1b) B(The x)Lxe.

Read this as the predicate 'B' applied to the "term" '(The x)Lxe'. 'The one who loves Eve' and '(The x)Lxe' are called *Definite Descriptions,* respectively in English and in logic. Traditionally, the definite description forming operator, (The **u**), is written with an upside-down Greek letter iota, '℩', like this: (℩**u**)**P**(**u**).

Here are some examples of definite descriptions transcribed into predicate logic:

a) The present king of France: (The x)Kx.
b) The blond son of Eve: (The x)(Bx & Sxe).
c) The one who loves all who love themselves: (The x)(∀y)(Lyy ⊃ Lxy).

But we can't treat (The x)**P**(x) like an ordinary term, because sometimes such "terms" don't refer. Consequently, we need a rewriting rule, just as we did for subscripted predicates and '(∃x!)', to show that expressions like (1b) should be rewritten as (1a):

> Rule for rewriting *Definite Descriptions Using '(The u)'*: **Q**[(The **u**)**P**(**u**)] is shorthand for (∃**u**!)[**P**(**u**) & **Q**(**u**)], where **P**(**u**) and **Q**(**u**) are open formulas with **u** the only free variable.

This treatment of definite descriptions works very smoothly, given the limitations of predicate logic. It does, however, introduce an oddity about the negations of sentences which use a definite description. How should we understand

(2) The one who loves Eve is not blond.

Anyone who holds a presupposition account will have no trouble with (2): They will say that if the presupposition holds, so that there is just one person who loves Eve, then (2) is true if the person is not blond and false if he or she is blond. If the presupposition fails, then (2), just as (1), is neither true nor false.

But what should we say in predicate logic about the transcription of (2)? We can see (2) as the negation of (1) in two very different ways. We can see (2) as the definite description '(The x)Lxe applied to the negated predicate '~B' in which case we have

(2a) ~B(The x)Lxe, rewritten as (∃x!)(Lxe & ~Bx).

When we think of (1) and (2) this way, we say that the definite description has *Primary Occurrence* or *Wide Scope.*

Or we can see (2) as the negation of the whole transcribed sentence:

(2b) ~[B(The x)Lxe], rewritten as ~(∃x!)(Lxe & Bx).

Thinking of (1) and (2) in this second way, we say that the definite description has *Secondary Occurrence* or *Narrow Scope.* When transcribing an English sentence with a definite description into logic, you will always have to make a choice between treating the definite description as having primary or secondary occurrence.

EXERCISES

Transcription Guide

a:	Adam	Dx:	x is dark-eyed
e:	Eve	Fxy:	x is a father of y
c:	Cain	Sxy:	x is a son of y
Bx:	x is blond	Cxy:	x is more clever than y
		Lxy:	x loves y

9–12. Transcribe the following. Expressions of the form (The **u**) and (∃**u**!) should not appear in your final answers.

a) The son of Eve is blond.
b) The son of Eve is more clever than Adam.
c) Adam is the father of Cain.
d) Adam loves the son of Eve.
e) Adam loves his son.
f) Cain loves the blond.
g) The paternal grandfather of Adam is dark-eyed.
h) The son of Eve is the son of Adam.
i) The blond is more clever than the dark-eyed one.
j) The most clever son of Adam is the father of Eve.
k) The son of the father of Eve is more clever than the father of the son of Adam.

9–13. Transcribe the negations of the sentences of exercise 9–12, once with the definite description having primary occurrence and once with secondary occurrence, indicating which transcription is which. Comment on how you think the notions of primary and secondary occurrence should work when a sentence has two definite descriptions.

CHAPTER SUMMARY EXERCISES

This chapter has introduced the following terms and ideas. Summarize them briefly.

a) Identity
b) Referent
c) Co-Referential
d) ($\exists$u!)
e) Self-Identity
f) Extensional
g) Extensional Semantics
h) Rule =I for Derivations
i) Rule =E for Derivations
j) Rule = for Trees
k) Rules ≠ for Trees
l) Reflexive Relation
m) Symmetric Relation
n) Transitive Relation
o) Equivalence Relation
p) Function
q) One Place Function
r) Two and Three Place Functions
s) Arguments of a Function
t) Function Symbols
u) Term
v) Constant, or Constant Term
w) Rules for Function Symbols in Derivations
x) Rules for Function Symbols in Trees
y) Presupposition
z) Definite Description
aa) Rewrite Rule for Definite Descriptions
bb) Primary Occurrence (Wide Scope) of a Definite Description
cc) Secondary Occurrence (Narrow Scope) of a Definite Description