Chapter **10**

PREDICATE LOGIC: DERIVATIONS

10.1 THE DERIVATION SYSTEM *PD*

In this chapter we develop natural deduction systems for predicate logic. The first system, *PD* (for *p*redicate *d*erivations), contains exactly two rules for each logical operator, just as *SD* contains exactly two rules for each sentential connective. It provides syntactic methods for evaluating sentences and sets of sentences of *PL,* just as the natural deduction system *SD* provides methods for evaluating sentences and sets of sentences of *SL*. *PD* is both complete and sound: for any set Γ of sentences of *PL* and any sentence **P** of *PL*

$\Gamma \vDash \mathbf{P}$ if and only if $\Gamma \vdash \mathbf{P}$ in *PD*.

That is, a sentence **P** of *PL* is quantificationally entailed by a set Γ of sentences of *PL* if and only if **P** is derivable from Γ in *PD*. We prove this in Chapter 11.

The derivation rules of *PD* include all the derivation rules of *SD,* with the understanding that they apply to sentences of *PL*. So the following is a derivation in *PD:*

Derive: ~ (∀x)Hx

1	(∀x)Hx ⊃ ~ (∃y)Py	Assumption
2	(∃y)Py	Assumption
3	(∀x)Hx	A / ~ I
4	~ (∃y)Py	1, 3 ⊃E
5	(∃y)Py	2 R
6	~ (∀x)Hx	3–5 ~ I

The strategies we used with *SD* are also useful when working in *PD*. Those strategies are based on careful analyses of the goal or goals of a derivation—the structure of the sentence or sentences to be derived—and the structure of accessible sentences. They can be summarized thus:

- If the current goal sentence can be obtained by Reiteration, use that rule, otherwise
- If the current goal sentence can be obtained by using a non-subderivation rule, or a series of such rules, do so; otherwise
- Try to obtain the goal sentence by using an appropriate subderivation rule.
- When using a negation rule, try to use an already accessible negation (if there is one) as the ~ **Q** that the negation rules require be derived.

The new rules of *PD* call for some new strategies. We will introduce these as we introduce the new derivation rules of *PD*. *PD* contains four new rules, Universal Elimination, Universal Introduction, Existential Elimination, and Existential Introduction. Each of the new rules involves a quantified sentence and a substitution instance of that sentence. The elimination rule for the universal quantifier is Universal Elimination:

Universal Elimination (∀E)

	(∀**x**)**P**
▷	**P**(**a**/**x**)

Here we use the expression '**P**(**a**/**x**)' to stand for a substitution instance of the quantified sentence (∀**x**)**P**. **P**(**a**/**x**) is obtained from the quantified sentence by dropping the initial quantifier and replacing every occurrence of **x** with **a**. We will refer to the constant **a** that is substituted for the variable **x** as the **instantiating constant** for the rule ∀E (and similarly for the other rules introduced on the following pages).

Universal Elimination allows us, given a universal generalization, to infer a sentence that says of a particular thing what the given universal generalization says of everything. Consider the following argument:

All philosophers are somewhat strange.
Socrates is a philosopher.

Socrates is somewhat strange.

The first premise makes a universal claim: it says that each thing is such that if it is a philosopher then it is somewhat strange. We can symbolize this claim as '(∀y)(Py ⊃ Sy)'. The second premise can be symbolized as 'Ps' and the conclusion as 'Ss'. Here is a derivation of the conclusion from the premises.

Derive: Ss

1	(∀y) (Py ⊃ Sy)	Assumption
2	Ps	Assumption
3	Ps ⊃ Ss	1 ∀E
4	Ss	2, 3 ⊃E

The sentence on line 3 is a substitution instance of the quantified sentence on line 1. When we remove the initial (and only) quantifier from '(∀y) (Py ⊃ Sy)' we get the open sentence 'Py ⊃ Sy', which contains two free occurrences of 'y'. Replacing both occurrences with the constant 's' yields the substitution instance 'Ps ⊃ Ss' on line 3, justified by ∀E. We then use Conditional Elimination to obtain 'Ss'.

This simple derivation illustrates the first new strategy for constructing derivations in *PD:*

- When using Universal Elimination use goal sentences as guides to which constant to use in forming the substitution instance of the universally quantified sentence.

At line 3 in the above derivation we could have entered 'Pa ⊃ Sa', or any other substitution instance of '(∀y) (Py ⊃ Sy)'. But obviously only the substitution instance using 's' is of any use in completing the derivation.

The instantiating constant employed in Universal Elimination may or may not already occur in the quantified sentence. The following is a correct use of Universal Elimination:

1	(∀x) Lxa	Assumption
2	Lta	1 ∀E

If we take our one assumption to symbolize 'Everyone loves Alice', with 'a' designating Alice, then clearly it follows that Tom, or whomever t designates, loves Alice. The following is also a correct use of Universal Elimination:

1	(∀x) Lxa	Assumption
2	Laa	1 ∀E

If everyone loves Alice, then it follows that Alice loves Alice, that is, that Alice loves herself.

The introduction rule for existential quantifiers is Existential Introduction:

Existential Introduction (∃I)

	P(**a**/**x**)
▷	(∃**x**)**P**

This rule allows us to infer an existentially quantified sentence from any one of its substitution instances. Here is an example:

1	Fa	Assumption
2	(∃y)Fy	1 ∃I

That Existential Introduction is truth-preserving should also be obvious. If the thing designated by the constant 'a' is F, then at least one thing is F. For example, if Alfred is a father, then it follows that someone is a father.

The following derivation uses Existential Introduction three times:

1	Faa	Assumption
2	(∃y)Fya	1 ∃I
3	(∃y)Fyy	1 ∃I
4	(∃y)Fay	1 ∃I

These uses are all correct because the sentence on line 1 is a substitution instance of the sentence on line 2, and of the sentence on line 3, and of the sentence on line 4. If Alice is fond of herself, then it follows that someone is fond of Alice, that someone is fond of her/himself, and that Alice is fond of someone.

The strategy for using Existential Introduction is straightforward:

- When the goal to be derived is an existentially quantified sentence establish a substitution instance of that sentence as a subgoal, with the intent of applying Existential Introduction to that subgoal to obtain the goal.

The rules Universal Introduction and Existential Elimination are somewhat more complicated. We begin with Universal Introduction:

Universal Introduction (∀I)

| **P**(**a**/**x**)

▷ | (∀**x**)**P**

provided that

(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in (∀**x**)**P**.

Here, again, we will call the constant **a** in **P**(**a**/**x**) the *instantiating constant*. This rule specifies that under certain conditions we can infer a universally quantified sentence from one of its substitution instances. At first glance this might seem implausible, for how can we infer, from a claim that a particular thing is of a certain sort, that *everything* is of that sort? The answer, of course, lies in the restrictions specified in the "provided that" clause.

Here is a very simple example. The sentences '(∀x)Fx' and '(∀y)Fy' are equivalent; they are simply notational variants of each other. They both say that everything is F. So we should be able to derive each from the other. Below we derive the second from the first:

Derive: (∀y)Fy

1	(∀x)Fx	Assumption
2	Fb	1 ∀E
3	(∀y)Fy	2 ∀I

The use of Universal Introduction at line 3 meets both the restrictions on that rule. The instantiating constant 'b' does not occur in an open assumption and does not occur in the universal generalization entered on line 3.

The kind of reasoning that Universal Introduction is based on is common in mathematics. Suppose we want to establish that no even positive integer greater than 2 is prime. [A prime is a positive integer that is evenly divisible only by itself and 1, and is not 1.] We might reason thus:

> Consider any even positive integer *i* greater than 2. Because *i* is even, *i* must be evenly divisible by 2. But since *i* is not 2 (it is greater than 2), it follows that *i* is evenly divisible by at least three positive integers: 1, 2, and *i* itself. So it is not the case that *i* is evenly divisible only by itself and 1, and *i* cannot be prime. Therefore no even positive integer greater than 2 is prime.

It would exhibit a misunderstanding of this reasoning to reply "but the positive integer *i* you considered might have been 4, and while the reasoning does hold of 4—it is not prime—that fact alone doesn't show that the reasoning holds of every even positive integer greater than 2. You haven't considered 6 and 8 and 10 and. . . ." It would be a misunderstanding because in saying 'Consider any even positive integer *i* greater than 2' we don't mean 'Pick one'. We say 'Consider any even positive integer *i* . . .' because it is easier to construct the argument when we are speaking, grammatically, in the singular ('*i* is . . .', '*i* is not . . .'). But what we are really saying is 'Consider what we know about all positive integers that are even and greater than two . . .' So the proof is a proof about all such integers. Similarly, in derivations we often use an individual constant to reason about all cases of a certain sort.

Suppose we want to establish that '(∀x)[Fx ⊃ (Fx ∨ Gx)]' can be derived from no assumptions. (This will, of course, establish that this sentence of *PL* is a theorem in *PD*.) Here is one such derivation:

Derive: (∀x)[Fx ⊃ (Fx ∨ Gx)]

1	Fc	A / ⊃I
2	Fc ∨ Gc	1 ∨I
3	Fc ⊃ (Fc ∨ Gc)	1–2 ⊃I
4	(∀x)[Fx ⊃ (Fx ∨ Gx)]	3 ∀I

The sentence on line 3 follows from the subderivation on lines 1–2, no matter what the constant ‘c’ designates. The subderivation establishes that no matter what c is, if it is F then it is F or G. Hence we are justified in deriving the universal quantification on line 4. Note that although ‘c’ occurs in the assumption on line 1, that assumption is not open at line 4, so we have not run afoul of the first restriction on the rule Universal Introduction.

On the other hand, Universal Introduction is misused in the following attempted derivation:

Derive: (∀y)Fy

1	Fb & ~ Fc	Assumption	
2	Fb	1 &E	
3	(∀y)Fy	2 ∀I	**MISTAKE!**

‘Fb’ does follow from line 1. But line 3 does not follow from line 2, and the restriction that the instantiating constant, in this case ‘b’, not occur in an open assumption prevents us from using Universal Introduction at line 3. (From the fact that Beth is a faculty member and Carl is not it does not follow that everyone is a faculty member.)

The rule Universal Introduction contains a second restriction, namely that the instantiating constant not occur in the derived universally quantified sentence. The following attempt at a derivation illustrates why this restriction is needed:

Derive: (∀x)Lxh

1	(∀x)Lxx	Assumption	
2	Lhh	1 ∀E	
3	(∀x)Lxh	2 ∀I	**MISTAKE!**

The sentence on line 1 tells us that everying bears L to itself. It certainly follows that h bears L to itself. But it does not follow that everything bears L to h, and the second restriction on Universal Introduction disallows the use of that rule to obtain the sentence on line 3, because the instantiating constant, ‘h’, does occur in that sentence.

The strategy associated with Universal Introduction is

- When the current goal is a universally quantified sentence make a substitution instance of that quantified sentence a subgoal, with the intent of applying Universal Introduction to derive the goal from the subgoal. Make sure that the two restrictions on Universal Introduction will be met: use an instantiating constant in the substitution instance that does not occur in the universally quantified goal sentence and that does not occur in any assumption that is open at the line where the substitution instance is entered.

Here is the elimination rule for existential quantifiers:

Existential Elimination (∃E)

	(∃**x**)**P**
	P(**a**/**x**)
	Q
▷	**Q**

provided that

(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in (∃**x**)**P**.
(iii) **a** does not occur in **Q**.

The idea behind this rule is that if we have an existentially quantified sentence (∃**x**)**P** then we know that something is of the sort specified by **P**, though not which thing. If, by assuming an arbitrary substitution instance **P**(**a**/**x**) of (∃**x**)**P**, we can derive a sentence **Q** that does not contain the instantiating constant **a** in **P**(**a**/**x**), then we can end the subderivation and enter **Q** on the next line of the derivation.

We illustrate a simple use of Existential Elimination by deriving '(∃x)(Gx ∨ Fx)' from {(∃z)Fz & (∀y)Hy}.

Derive: (∃x)(Gx ∨ Fx)

1	(∃z)Fz & (∀y)Hy	Assumption
2	(∃z)Fz	1 &E
3	Fb	A / ∃E
4	Gb ∨ Fb	3 ∨I
5	(∃x)(Gx ∨ Fx)	4 ∃I
6	(∃x)(Gx ∨ Fx)	2, 3–5 ∃E

'Existential Elimination' may seem like an odd name for the rule we used at line 6 of the above derivation, because the sentence entered at line 6 is itself an existentially quantified sentence. But remember that what is common to all elimination rules is that they are rules that *start* with a sentence with a specified main logical operator and produce a sentence that may or may not have that operator as a main logical operator. Here Existential Elimination cites the existentially quantified sentence at line 2, along with the subderivation beginning with a substitution instance of that sentence. Note that we have met all the restrictions on Existential Elimination. The instantiating constant 'b' does not occur in an assumption that is open as of line 6. Nor does 'b' occur in '(∃z)Fz'. Finally, 'b' does not occur in the sentence that is derived, at line 6, by Existential Elimination. All three of these restrictions are necessary, as we will now illustrate.

Two specific strategies are associated with the rule Existential Elimination. The first is this:

- When one or more of the currently accessible sentences in a derivation is an existentially quantified sentence, consider using Existential Elimination to obtain the current goal. Assume a substitution instance that contains a constant that does not occur in the existential quantification, in an open assumption, or in the current goal. Work within the Existential Elimination subderivation to derive the current goal.

In other words, whenever an existentially quantified sentence is accessible consider making Existential Elimination the primary strategy for obtaining the current goal, doing the work required to obtain the current goal within the scope of the Existential Elimination subderivation. This is often necessary to avoid violating the restrictions on Existential Elimination. For example, in the previous derivation we had to use Existential Introduction within the scope of the assumption on line 3—because trying to derive 'Gb ∨ Fb' by Existential Elimination at line 5, prior to applying Existential Introduction, would violate the third restriction on Existential Elimination:

Derive: (∃x) (Gx ∨ Fx)

1	(∃z)Fz & (∀y)Hy	Assumption	
2	(∃z)Fz	1 &E	
3	Fb	A / ∃E	
4	Gb ∨ Fb	3 ∨I	
5	Gb ∨ Fb	2, 3–4 ∃E	**MISTAKE!**
6	(∃x) (Gx ∨ Fx)	5 ∃I	

Line 5 is a mistake because the instantiating constant 'b' occurs in the sentence we are trying to obtain by Existential Elimination, in violation of the third restriction on Existential Elimination. From the truth of '(∃z)Fz' it does not follow that the individual designated by 'b' is either G or F—although it does follow, as in the previous derivation, that *something* is either G or F. This is why, in the correctly done derivation, we used Existential Introduction inside of the Existential Elimination subderivation. Doing so results in a sentence that does not contain the instantiating constant 'b' and that therefore can correctly be moved out of the subderivation by Existential Elimination.

Here is another example in which the third restriction on Existential Elimination is violated:

Derive: (∃z)Fbz

1	(∃z)Fzz	Assumption	
2	Fbb	A / ∃E	
3	(∃z)Fbz	3 ∃I	
4	(∃z)Fbz	1, 2–3 ∃E	**MISTAKE!**

The instantiating constant ‘b’ occurs in the sentence on line 4, in violation of restriction (iii) on Existential Elimination. It is clear that we don’t want the above to count as a derivation. Given the assumption on line 1 we know that something bears F to itself. At line 2 we assume that thing is b (knowing that this may not be the case). Line 3 certainly follows from line 2. *If* b bears F to itself then b does bear F to something. But line 4, where we have given up the assumption that it is b that bears F to itself, does not follow from the sentence at line 1, which is the single open assumption as of line 4. Contrast the preceding derivation with the following:

Derive: (∃y)Fyy

1	(∃z)Fzz	Assumption
2	Fbb	A / ∃E
3	(∃y)Fyy	2 ∃I
4	(∃y)Fyy	1, 2–3 ∃E

Here ‘b’ does not occur in the sentence at line 4, so the third restriction on Existential Elimination is not violated. We have used Existential Elimination to show that ‘(∃y)Fyy’ follows from ‘(∃z)Fzz’, which should be no surprise since these sentences are clearly equivalent.

We will now examine some misuses of Existential Elimination that illustrate why the two other restrictions on Existential Elimination are also necessary.

Derive: (∀x)Fx

1	Gb ⊃ (∀x)Fx	Assumption	
2	(∃z)Gz	Assumption	
3	Gb	A / ∃E	
4	(∀x)Fx	1, 3 ⊃E	
5	(∀x)Fx	2, 3–4 ∃E	**MISTAKE!**

From line 1 we know that if a particular thing, namely b, is G, then everything is F. And from line 2 we know that something is G. But we do not know that it is b that is G. So we should not be able to infer, as we have here tried to do at line 5, that everything is F. Line 5 is a mistaken application of Existential Elimination because restriction (i) has not been met. The assumption at line 1, which contains the instantiating constant ‘b’, is still open as of line 5. The rationale for restriction (i) should now be clear. Existential Elimination uses a substitution instance of an existentially quantified claim to show what follows from the existentially quantified claim. But the constant used in the substitution instance, the instantiating constant, should be arbitrary, in the sense that no assumptions have been made concerning the thing designated by that constant. If the instantiating constant occurs in an open assumption then it is not arbitrarily selected, because the open assumption provides information about b (that if it is G everything is F). It may be the case that if Bob graduates then everyone is happy and the case that someone does graduate. But it does not follow from this that everyone is happy, for the someone who graduates may not be Bob.

We now turn to the rationale for the second restriction. Consider the following attempt at a derivation:

Derive: (∃w)Lww

1	(∀y)(∃z)Lzy	Assumption	
2	(∃z)Lza	1 ∀E	
3	Laa	A / ∃E	
4	(∃w)Lww	3 ∃I	
5	(∃w)Lww	2, 3–4 ∃E	**MISTAKE!**

The problem is that the instantiating constant 'a' used at line 3 to form a substitution instance of the sentence '(∃z)Lza' occurs in '(∃x)Lza', violating the second restriction. If we only know that *something* stands in the relation L to a, we should not assume that that something is in fact a itself.

Universal Elimination produces a substitution of the universally quantified sentence to which it is applied. Existential Elimination does not, in general, produce a substitution instance of the existentially quantified sentence to which it is applied. Indeed the sentence it produces may bear no resemblance, by any normal standard of resemblance, to the existentially quantified sentence to which it is applied. Here is a case in point:

Derive: (∃x)Hx

1	(∃z)Gz	Assumption
2	(∀y)(Gy ⊃ Hc)	Assumption
3	Gb	A / ∃E
4	Gb ⊃ Hc	2 ∀E
5	Hc	3, 4 ⊃E
6	Hc	1, 3–5 ∃E
7	(∃x)Hx	6 ∃I

Here the sentence derived at line 6 has no obvious connection to the existentially quantified sentence at line 1. The existentially quantified sentence tells us that something is G. At line 3 we assume that that thing is b. The constant 'b' is not used earlier in the derivation, so we are committed to nothing about b other than its being G. At line 4 we use Universal Elimination to obtain 'Gb ⊃ Hc', and then we use Conditional Elimination at line 5 to obtain 'Hc'. At the point we apply Existential Elimination (line 6) there is here no open assumption that contains 'b'—the only open assumptions are those on line 1 and line 2—so the first restriction on Existential Elimination is met. The second and third restrictions are also met since 'b' occurs in neither '(∃z)Gz' nor 'Hc'. We can, therefore, derive 'Hc' by Existential Elimination at line 6.

Note that in this case we were able to move 'Hc' out of the Existential Elimination subderivation prior to using Existential Introduction. We could do this because 'c' was not the instantiating constant for our use of Existential Elimination. However, we could also have applied Existential Introduction within the subderivation;

Derive: (∃x)Hx

1	(∃z)Gz	Assumption
2	(∀y)(Gy ⊃ Hc)	Assumption
3	Gb	A / ∃E
4	Gb ⊃ Hc	2 ∀E
5	Hc	3, 4 ⊃E
6	(∃x)Hx	5 ∃I
7	(∃x)Hx	1, 3–6 ∃E

Existential Elimination provides a strategy for working from a substitution instance of an existentially quantified sentence to a sentence that does not contain the instantiating constant of the substitution instance. If the other restrictions on Existential Elimination are also met the subderivation can be ended and the derived sentence entered on the next line of the derivation.

There is a second important strategy associated with Existential Elimination. We will use it to show that the set {(∃x) ~ Fx, (∀x)Fx} is inconsistent in *PD*. The foregoing set obviously is inconsistent, but demonstrating this is not as easy as it might seem. We might start as follows:

Derive: Fa, ~ Fa

1	(∃x) ~ Fx	Assumption	
2	(∀x)Fx	Assumption	
3	Fa	2 ∀E	
4	~ Fa	1 ∃E	**MISTAKE!**

Line 4 is an obvious misuse of Existential Elimination. A more promising approach might be as follows:

Derive: Fa, ~ Fa

1	(∃x) ~ Fx	Assumption
2	(∀x)Fx	Assumption
3	~ Fa	A / ∃E
4	Fa	2 ∀E
5	~ Fa	3 R

We have derived a sentence and its negation ('Fa' and '~ Fa'), but only within the scope of our Existential Elimination subderivation. And since 'a' is the instantiating constant of the assumption at line 3, we cannot hope to move either 'Fa' or '~ Fa' out from the scope of the assumption at line 3 by Existential Elimination. The situation we are in is not an uncommon one. We need to use Existential Elimination, and we can derive a contradiction within the Existential Elimination subderivation, but the contradictory sentences we derive cannot be moved outside that subderivation because they contain the instantiating constant of the assumption.

The strategy we will use in situations such as this makes use of the fact that we can derive contradictory sentences within the Existential Elimination subderivation. Since we can do this, we can also derive any sentence we want by use of the appropriate negation rule. In our present case we want to derive a sentence and its negation, to show that the set we are working from is inconsistent in *PD*. There are no negations among our primary assumptions. We know taking 'Fa' and '~ Fa' as our ultimate goals will not work (so long as '~ Fa' remains as our Existential Elimination assumption at line 3). So we will take a sentence that is accessable, '(∀x)Fx', and its negation as our ultimate goals, and we will derive '~ (∀x)Fx' by Negation Introduction within our Existential Elimination subderivation, and then move it out of that subderivation by Existential Elimination:

Derive: (∀x)Fx, ~ (∀x)Fx

1	| (∃x) ~ Fx	Assumption
2	| (∀x)Fx	Assumption
3	| | ~ Fa	A / ∃E
4	| | | (∀x)Fx	A / ~ I
5	| | | Fa	4 ∀E
6	| | | ~ Fa	3 R
7	| | ~ (∀x)Fx	4–6 ~ I
8	| ~ (∀x)Fx	1, 3–7 ∃E
9	| (∀x)Fx	2 R

What may strike one as odd about this derivation is that we are assuming, at line 4, a sentence that is already accessible (as the assumption on line 2). But the point of making this assumption of a sentence we already have is to derive its negation, which we do at line 7. Negation Introduction requires us to assume this sentence, even though it also occurs at line 2, before we can apply that rule. At line 4 we could, of course, have equally well assumed '(∃x) ~ Fx', in which case our ultimate goals would have been '(∃x) ~ Fx' and '~ (∃x) ~ Fx'.

The strategy we are illustrating can be put thus:

- When contradictory sentences are available within an Existential Elimination subderivation but cannot be moved out of that subderivation without violating the restrictions on Existential Elimination, derive another sentence—one that is contradictory to a sentence accessible outside the Existential Elimination subderivation and one that can be moved out. That sentence will be derivable by the appropriate negation strategy (because contradictory sentences are available within the Existential Elimination subderivation).

Using this strategy will frequently involve assuming, as the assumption of a negation strategy, a sentence that is already accessible outside the Existential Elimination subderivation.

Consider next the following failed attempt at a derivation of '~ (∃x)Fx' from {~ (∃x)Fx}:

```
Derive: ~ (∃x)Fx

1  | (∀x) ~ Fx           Assumption
   |----------
2  |  | (∃x)Fx           A / ~ I
   |  |-------
3  |  |  | Fa            A / ∃E
   |  |  |---
4  |  |  | ~ Fa          1 ∀E
5  |  |  | Fa            3 R
6  |  | ~ (∃x)Fx         3–5 ~ I          MISTAKE!
7  | ~ (∃x)Fx            2, 3–6 ∃E        MISTAKE!
```

We are trying to derive a negation, '~ (∃x)Fx', and so assume '(∃x)Fx' at line 2. Clearly an Existential Elimination strategy is now called for, and accordingly we assume 'Fa' at line 3. It is now easy to derive the contradictory sentences 'Fa' and '~ Fa', and we do so at lines 4 and 5. But line 6 is a mistake. Our primary strategy is Negation Introduction and we have derived a sentence and its negation; but we have done so only within the scope of an additional assumption, the one at line 3 that begins our Existential Elimination strategy. Line 6 is a mistake because 'Fa' and '~ Fa' have been derived, not from just the assumptions on lines 1 and 2, but also using the assumption on line 3. We need to complete our Existential Elimination strategy before using Negation Introduction. And what we want our Existential Elimination strategy to yield is a sentence that can serve as one of the contradictory sentences we need to complete the Negation Introduction subderivation we began at line 2.

Two sentences are accessible outside our Existential Elimination subderivation—those on lines 1 and 2 ('(∀x) ~ Fx' and '(∃x)Fx') and obtaining the negation of either one of these by Existental Elimination will allow us to complete the derivation. Here is a successful derivation in which we derive '~ (∀x) ~ Fx' by Existential Elimination.

```
Derive: ~ (∃x)Fx

1   | (∀x) ~ Fx                Assumption
    |----------
2   |  | (∃x)Fx                A / ~ I
    |  |-------
3   |  |  | Fa                 A / ∃E
    |  |  |---
4   |  |  |  | (∀x) ~ Fx       A / ~ I
    |  |  |  |----------
5   |  |  |  | ~ Fa            1 ∀E
6   |  |  |  | Fa              3 R
7   |  |  | ~ (∀x) ~ Fx        4–6 ~ I
8   |  | ~ (∀x) ~ Fx           2, 3–7 ∃E
9   |  | (∀x) ~ Fx             1 R
10  | ~ (∃x)Fx                 2–9 ~ I
```

After making the assumption at line 3 we realize we can derive the contradictory sentences 'Fa' and '~ Fa'. Because we want to obtain '~ (∀x) ~ Fx' by Existential Elimination, we assume '(∀x) ~ Fx' at line 4 and derive '~ Fa' and 'Fa' within the scope of that assumption, allowing us to then derive '~ (∀x) ~ Fx' by Negation Introduction.

Alternatively, we could have used '(∃x)Fx' as an assumption at line 4, derived 'Fa' and '~ Fa', obtained '~ (∃x)Fx' by Negation Elimination, moved that sentence out of the scope of the assumption made at line 3 by Existential Elimination, and then reiterated '(∃x)Fx' within the scope of the assumption '(∃x)Fx' so as to have the contradictory sentences we need to finish the derivation with Negation Introduction. Note also that the assumption at line 4 is necessary to obtain its negation even though the sentence we assume is already available as an earlier assumption (on line 1). As noted earlier this process of making an assumption of a sentence that is already available outside the scope of an Existential Elimination strategy within that strategy *in order to obtain its negation* is extremely useful and frequently called for, as we will see in examples and exercises later in this chapter.

As another example, suppose we want to derive '~ (∃x) (Fx & ~ Gx)' from {(∀x) (~ Gx ⊃ ~ Fx)}. Since our primary goal is a negation, we plan to use Negation Introduction, and since the assumption of that strategy will be an existentially quantified sentence, we will use Existential Elimination within the Negation Introduction subderivation:

Derive: ~ (∃x) (Fx & ~ Gx)

1	(∀x) (~ Gx ⊃ ~ Fx)	Assumption
2	| (∃x) (Fx & ~ Gx)	A / ~ I
3	| | Fa & ~ Ga	A / ∃E
G	~ (∃x) (Fx & ~ Gx)	

Following our new strategy we will begin a Negation Introduction subderivation inside of the Existential Elimination subderivation, assuming one of the sentences that is accessible from outside of that subderivation. In this example there are again two such sentences, '(∀x) (~ Gx ⊃ ~ Fx)' and '(∃x) (Fx & ~ Gx)'. We arbitrarily select the latter as the assumption of the inner Negation Introduction subderivation and complete the derivation as follows:

```
     Derive: ~ (∃x)(Fx & ~ Gx)

 1 │ (∀x)(~ Gx ⊃ ~ Fx)                    Assumption
   ├──────────────────
 2 │  │ (∃x)(Fx & ~ Gx)                   A / ~ I
   │  ├────────────────
 3 │  │  │ Fa & ~ Ga                      A / ∃E
   │  │  ├──────────
 4 │  │  │  │ (∃x)(Fx & ~ Gx)             A / ~ I
   │  │  │  ├────────────────
 5 │  │  │  │ ~ Ga ⊃ ~ Fa                 1 ∀E
 6 │  │  │  │ ~ Ga                        3 &E
 7 │  │  │  │ ~ Fa                        5, 6 ⊃E
 8 │  │  │  │ Fa                          3 &E
 9 │  │  │ ~ (∃x)(Fx & ~ Gx)              4–8 ~ I
10 │  │ ~ (∃x)(Fx & ~ Gx)                 2, 3–9 ∃E
11 │  │ (∃x)(Fx & ~ Gx)                   2 R
12 │ ~ (∃x)(Fx & ~ Gx)                    2–11 ~ I
```

Although the assumption at line 4 is an existentially quantified sentence, there is no need for a second use of Existential Elimination. We can derive the contradictory pair of sentences 'Fa' and '~ Fa' without making any additional assumptions.

We have specified strategies for using each of the four new quantifier rules. Now that we have introduced all the rules of *PD* a note about applying those rules is in order. The quantifier introduction and elimination rules, like all the rules of *PD,* are *rules of inference.* That is, they apply only to whole sentences, *not* to subsentential components of sentences that may or may not themselves be sentences. The only sentences that quantifier elimination rules can be applied to are sentences whose main logical operators are quantifiers. Moreover, the quantifier *introduction* rules generate only sentences whose main logical operators are quantifiers. The following examples illustrate some common types of mistakes that ignore these points about the quantifier rules of *PD.*

```
     Derive: Fa ⊃ Ha

 1 │ (∀x)Fx ⊃ Ha              Assumption
   ├─────────────
 2 │ Fa ⊃ Ha                  1 ∀E                MISTAKE!
```

The sentence on line 1 is not a universally quantified sentence. Rather, it is a material conditional, so Universal Elimination cannot be applied to it. Obviously, the sentence on line 2 does not follow from the sentence on line 1. From that fact that if everything is F then a is H it does not follow that if a (which is only one thing) is F then a is H.

Here is another example illustrating a similar mistake:

Derive: Ga

1	Fa	Assumption	
2	(∀x)(Fx ⊃ (∀y)Gy)	Assumption	
3	(∀x)(Fx ⊃ Ga)	2 ∀E	**MISTAKE!**
4	Fa ⊃ Ga	3 ∀E	
5	Ga	1, 4 ⊃E	

Line 3 is a mistake even though the sentence it cites, '(∀x)(Fx ⊃ (∀y)Gy)', is a universally quantified sentence. It is a mistake because it attempts to apply Universal Elimination to '(∀y)Gy', which occurs only as a component of the sentence on line 2. Rules of inference can only be applied to sentences that are not components of larger sentences. Universal Elimination can only produce a substitution instance, for example 'Fa ⊃ (∀y)Gy', of the entire sentence on line 2.

We hasten to add that it *is* possible to derive 'Ga' from the sentences on lines 1 and 2 but a different strategy is required:

Derive: Ga

1	Fa	Assumption
2	(∀x)(Fx ⊃ (∀y)Gy)	Assumption
3	Fa ⊃ (∀y)Gy	2 ∀E
4	(∀y)Gy	1, 3 ⊃E
5	Ga	4 ∀E

Here Universal Elimination has only been applied to entire sentences occurring on earlier lines.

The following also illustrates a misuse of a quantifier rule:

Derive: (∃z)Fz ⊃ Gb

1	Fa ⊃ Gb	Assumption	
2	(∃z)Fz ⊃ Gb	1 ∃I	**MISTAKE!**

Existential Introduction produces existentially quantified sentences, and the sentence on line 2 is a material conditional, not an existentially quantified sentence. Nor do we want to be able to derive the sentence on line 2 from the sentence on line 1. From 'If Alfred wins the election then Bob will be happy' it does not follow that if *someone* wins the election then Bob will be happy. A correct use of the rule would be

Derive: (∃z)(Fz ⊃ Gb)

1	Fa ⊃ Gb	Assumption
2	(∃z)(Fz ⊃ Gb)	1 ∃I

In the following failed derivation, the use of Universal Elimination is incorrect because the sentence on line 1 is not a universally quantified sentence. Rather, it is the *negation* of a universally quantified sentence:

Derive: $\sim Fb$

1	$\sim (\forall y)Fy$	Assumption	
2	$\sim Fb$	1 $\forall$E	**MISTAKE!**

Having introduced all the rules of *PD* we can now define syntactic analogues of core logical concepts for *PD:*

> *Derivability in PD:* A sentence **P** of *PL* is *derivable in PD* from a set Γ of sentences of *PL* if and only if there is a derivation in *PD* in which all the primary assumptions are members of Γ and **P** occurs within the scope of only the primary assumptions.
>
> *Validity in PD:* An argument of *PL* is *valid in PD* if and only if the conclusion of the argument is derivable in *PD* from the set consisting of the premises. An argument of *PL* is *invalid in PD* if and only if it is not valid in *PD*.
>
> *Theorem in PD:* A sentence **P** of *PL* is a *theorem in PD* if and only if **P** is derivable in *PD* from the empty set.
>
> *Equivalence in PD:* Sentences **P** and **Q** of *PL* are *equivalent in PD* if and only if **Q** is derivable in *PD* from {**P**} and **P** is derivable in *PD* from {**Q**}.
>
> *Inconsistency in PD:* A set Γ of sentences of *PL* is *inconsistent in PD* if and only if there is a sentence **P** such that both **P** and ~ **P** are derivable in *PD* from Γ. A set Γ is *consistent in PD* if and only if it is not inconsistent in *PD*.

10.1E EXERCISES

1. Construct derivations that establish the following claims:

a. $\{(\forall x)Fx\} \vdash (\forall y)Fy$
*b. $\{Fb, Gb\} \vdash (\exists x)(Fx \mathbin{\&} Gx)$
c. $\{(\forall x)(\forall y)Hxy\} \vdash (\exists x)(\exists y)Hxy$
*d. $\{(\exists x)(Fx \mathbin{\&} Gx)\} \vdash (\exists y)Fy \mathbin{\&} (\exists w)Gw$
e. $\{(\forall x)(\forall y)Hxy, Hab \supset Kg\} \vdash Kg$
*f. $\{(\forall x)(Fx \equiv Gx), (\forall y)(Gy \equiv Hy)\} \vdash (\forall x)(Fx \equiv Hx)$
g. $\{(\forall x)Sx, (\exists y)Sy \supset (\forall w)Ww\} \vdash (\exists y)Wy$
*h. $\{(\forall y)Hyy, (\exists z)Bz\} \vdash (\exists x)(Bx \mathbin{\&} Hxx)$
i. $\{(\forall x)(\forall y)Lxy, (\exists w)Hww\} \vdash (\exists x)(Lxx \mathbin{\&} Hxx)$
*j. $\{(\forall x)(Fx \supset Lx), (\exists y)Fy\} \vdash (\exists x)Lx$

2. Identify the mistake in each of the following attempted derivations, and explain why it is a mistake.

a. Derive: Na

1	$(\forall x)Hx \supset \sim (\exists y)Ky$	Assumption
2	$Ha \supset Na$	Assumption
	—	
3	Ha	1 $\forall$E
4	Na	2, 3 $\supset$E

*b. Derive: $(\forall x)(Bx \;\&\; Mx)$

1	Bk	Assumption
2	$(\forall x)Mx$	Assumption
	—	
3	Mk	2 $\forall$E
4	$Bk \;\&\; Mk$	1, 3 &I
5	$(\forall x)(Bx \;\&\; Mx)$	4 $\forall$I

c. Derive: $(\exists x)Cx$

1	$(\exists y)Fy$	Assumption
2	$(\forall w)(Fw \equiv Cw)$	Assumption
	—	
3	Fa	1 $\exists$E
4	$Fa \equiv Ca$	2 $\forall\exists$E
5	Ca	3, 4 $\equiv$E
6	$(\exists x)Cx$	5 $\exists$I

*d Derive: $(\exists z)Gz$

1	$(\forall x)(Fx \supset Gx)$	Assumption
2	$(\exists y)Fy$	Assumption
	—	
3	│ Fa	A / $\exists$E
	│ —	
4	│ $Fa \supset Ga$	1 $\forall$E
5	│ Ga	3, 4 $\supset$E
6	Ga	2, 3–5 $\exists$E
7	$(\exists z)Gz$	6 $\exists$I

e. Derive: $(\exists y)(\forall x)Ayx$

1	$(\forall x)(\exists y)Ayx$	Assumption
	—	
2	$(\forall x)Aax$	1 $\forall$E
3	$(\exists y)(\forall x)Ayx$	2 $\exists$I

*f. Derive: $\sim Rba$

1	$(\exists x)Rxx$	Assumption
2	$(\forall x)(\forall y)(Rxy \supset \sim Ryx)$	Assumption
	—	
3	│ Raa	A / $\exists$E
	│ —	
4	│ $(\forall y)(Ray \supset \sim Rya)$	2 $\forall$E
5	│ $Raa \supset \sim Raa$	2 $\forall$E
6	│ $\sim Raa$	3, 5 $\supset$E
7	│ $(\forall x) \sim Rxx$	6 $\forall$I
8	$(\forall x) \sim Rxx$	1, 3–7 $\exists$E

10.2 USING DERIVATIONS TO ESTABLISH SYNTACTIC PROPERTIES OF *PD*

In this section we will work through a series of derivations, illustrating both strategies that are useful in constructing derivations in *PD* and how derivations are used to establish that various syntactic properties of *PD* hold of sentences and sets of sentences of *PL*.

We begin by repeating the strategies we have enumerated as useful in constructing derivations:

- If the current goal sentence can be obtained by Reiteration, use that rule, otherwise
- If the current goal sentence can be obtained by using a non-subderivation rule, or a series of such rules, do so; otherwise
- Try to obtain the goal sentence by using an appropriate subderivation rule.
- When using a negation rule, try to use an already accessible negation (if there is one) as the ~ **Q** that the negation rules require be derived.
- When using Universal Elimination use goal sentences as guides when choosing the instantiating constant.
- When the goal to be derived is an existentially quantified sentence make a substitution instance of that sentence a subgoal, with the intent of applying Existential Introduction to that subgoal to obtain the goal.
- When the current goal is a universally quantified sentence make a substitution instance of that quantified sentence a subgoal, with the intent of applying Universal Introduction to that subgoal. Make sure the two restrictions on the instantiating constant for the use of Universal Introduction are met. Be sure to choose an instantiating constant that does not occur in the universally quantified sentence that is the goal and that does not occur in any assumption that will be open when Universal Introduction is applied to derive that goal.
- When one of the accessible assumptions is an existentially quantified sentence, consider using Existential Elimination to obtain the current goal. Set up an Existential Elimination subderivation, and continue working within that subderivation until a sentence that does not contain the constant used to form the substitution instance that is the assumption of that subderivation is derived.
- When contradictory sentences are available within an Existential Elimination subderivation but cannot be moved out of that subderivation without violating the restrictions on Existential Elimination, derive another sentence—one that is contradictory to a sentence

accessible outside the Existential Elimination subderivation and that does not contain the instantiating constant for this use of Existential Elimination. That sentence will be derivable by the appropriate negation strategy (using the contradictory sentences that are available within the Existential Elimination subderivation).

- There will often be more than one plausible strategy, and often more than one will lead to success. Rather than trying to figure out which of these is the most promising it is often wise to just pick one and pursue it.

ARGUMENTS

An argument of *PL* is valid in *PD* if and only if the conclusion can be derived from the set consisting of the argument's premises. The following argument is valid in *PD,* as we will now show.

(∃x)(Fx & Gx)

(∃y)Fy & (∃z)Gz

The single premise is an existentially quantified sentence—which suggests using Existential Elimination. The conclusion is a conjunction, suggesting Conjunction Introduction as a strategy. We will use both strategies, and since it is in general wise to do as much work as possible within an Existential Elimination strategy (so as to avoid violating the third restriction on Existential Elimination), we will make that strategy our primary strategy. We begin as follows:

```
Derive: (∃y)Fy & (∃z)Gz

1  | (∃x)(Fx & Gx)            Assumption
   |---------------
2  | | Fb & Gb                A / ∃E
   | |---------
   | |
   | |
G  | | (∃y)Fy & (∃z)Gz        —, — &I
G  | (∃y)Fy & (∃z)Gz          1, 2–— ∃E
```

We will try to derive the conclusion of the argument *within* the scope of the Existential Elimination subderivation because doing so will avoid violating the third restriction on Existential Elimination, that the instantiating constant not occur in the derived sentence. In our derivation 'a' is the instantiating constant and it does not occur in the conclusion of the argument.

Our current goal is a conjunction and can be obtained by Conjunction Introduction. The completed derivation is

```
Derive: (∃y)Fy & (∃z)Gz

1 | (∃x)(Fx & Gx)              Assumption
  |--------------
2 |  | Fa & Ga                 A / ∃E
  |  |--------
3 |  | Fa                      2 &E
4 |  | (∃y)Fy                  3 ∃I
5 |  | Ga                      2 &E
6 |  | (∃z)Gz                  5 ∃I
7 |  | (∃y)Fy & (∃z)Gz         4, 6 &I
8 | (∃y)Fy & (∃z)Gz            1, 2–7 ∃E
```

The following argument is also valid in *PD:*

```
(∀x)(Nx ⊃ Ox)
~ (∃y)Oy
-----------
~ (∃x)Nx
```

Since the conclusion of this argument is a negation we will use Negation Introduction as our primary strategy and we will try to derive both '(∃y)Oy' and '~ (∃y)Oy' within a Negation Introduction subderivation:

```
Derive: ~ (∃x)Nx

1 | (∀x)(Nx ⊃ Ox)              Assumption
2 | ~ (∃y)Oy                   Assumption
  |--------------
3 |  | (∃x)Nx                  A / ~ E
  |  |-------
  |  |
  |  |
G |  | (∃y)Oy
  |  | ~ (∃y)Oy                2 R
G | ~ (∃x)Nx                   3–__ ~ I
```

Since one of the accessible sentences, '(∃x)Nx' is an existentially quantified sentence, we will try to obtain our current goal, '(∃y)Oy', by Existential Elimination:

```
Derive: ~ (∃x)Nx

1 | (∀x)(Nx ⊃ Ox)              Assumption
2 | ~ (∃y)Oy                   Assumption
  |--------------
3 |  | (∃x)Nx                  A / ~ E
  |  |-------
4 |  |  | Na                   A / ∃E
  |  |  |---
  |  |  |
G |  |  | (∃y)Oy
G |  | (∃y)Oy                  3, 4–__ ∃E
G |  | ~ (∃y)Oy                2 R
G | ~ (∃x)Nx                   3–__ ~ I
```

Looking at the sentences on lines 1 and 4, we see that we will be able to derive ‘Oa’ by Conditional Elimination after applying Universal Elimination to the sentence on line 1, with ‘a’ as the instantiating constant. And from ‘Oa’ we can obtain ‘(∃y)Oy’ by Existential Introduction. So the completed derivation is

Derive: ~ (∃x)Nx

1	(∀x)(Nx ⊃ Ox)	Assumption
2	~ (∃y)Oy	Assumption
3	(∃x)Nx	A / ~ E
4	Na	A / ∃E
5	Na ⊃ Oa	1 ∀E
6	Oa	4, 5 ⊃E
7	(∃y)Oy	6 ∃I
8	(∃y)Oy	3, 4–7 ∃E
9	~ (∃y)Oy	2 R
10	~ (∃x)Nx	3–9 ~ I

We will next consider two arguments, both of which involve relational predicates and quantifiers with overlapping scope. The first is

(∀x)(∀y)(Hxy ⊃ ~ Hyx)
(∀x)(∃y)Hxy
———————————
(∀x)(∃y) ~ Hxy

Here our assumptions and our goal sentence are all universally quantified sentences. So we will clearly be using Universal Elimination and Universal Introduction. Using Universal elimination on the second premise will result in an existentially quantified sentence, ‘(∃y)Hay’, which suggests using Existential Elimination:

Derive: (∀x)(∃y) ~ Hxy

1	(∀x)(∀y)(Hxy ⊃ ~ Hyx)	Assumption
2	(∀x)(∃y)Hxy	Assumption
3	(∃y)Hay	2 ∀E
4	Hab	A / ∃E
G	(∀x)(∃y) ~ Hyx	3, 4–__ ∃E

On line 4 we chose an instantiating constant that does not appear earlier in the derivation, so that the restrictions on the instantiating constant can be met. Clearly at some point we will obtain '(∀x)(∃y) ~ Hyx' by Universal Introduction. The question is whether we will use Universal Introduction before or after ending our Existential Elimination subderivation. We have stressed in earlier examples that it is generally wise to do as much work as possible within Existential Elimination subderivations. This might suggest that we try to obtain '(∀x)(∃y) ~ Hyx' within our Existential Elimination subderivation. But this is, in the present context, a bad idea. The substitution instance of '(∀x)(∃y) ~ Hyx' we will be able to obtain is '(∃y) ~ Hya', in which 'a' is the instantiating constant. The first restriction on Universal Introduction requires that the instantiating constant not occur in any open assumption. But 'a' does occur in 'Hab', the assumption on line 4. So we cannot apply Universal Introduction within the scope of that assumption.

A strategy that will work is to obtain '(∃y) ~ Hya' by Existential Elimination and then, after the assumption 'Hab' is discharged, to apply Universal Introduction. Note that our advice—to do as much work within Existential Elimination subderivations as possible—still holds. The current case is simply a reminder that doing as much work as possible within an Existential Elimination subderivation means, in part, doing as much work as can be done without violating the restrictions on the rules we use.

We have now settled on the following strategy:

Derive: (∀x)(∃y) ~ Hxy

1	(∀x)(∀y)(Hxy ⊃ ~ Hyx)	Assumption
2	(∀x)(∃y)Hxy	Assumption
3	(∃y)Hay	2 ∀E
4	Hab	A / ∃E
G	(∃y) ~ Hya	
G	(∃y) ~ Hya	3, 4–__ ∃E
G	(∀x)(∃y) ~ Hyx	__ ∀I

Our current goal is '(∃y) ~ Hya'. We would like to use Existential Introduction to derive this sentence, which means we first have to derive a substitution instance of this sentence. Looking at our first assumption, '(∀x)(∀y)(Hxy ⊃ ~ Hyx)', we see that with two applications of Universal Elimination we can obtain 'Hab ⊃ ~ Hba', then we can use Conditional Elimination to derive '~ Hba', a substitution instance of our goal, '(∃y) ~ Hya'. Our completed derivation is

Derive: (∀x)(∃y) ~ Hyx

1	(∀x)(∀y)(Hxy ⊃ ~ Hyx)	Assumption
2	(∀x)(∃y)Hxy	Assumption
3	(∃y)Hay	2 ∀E
4	Hab	A / ∃E
5	(∀y)(Hay ⊃ ~ Hya)	1 ∀E
6	Hab ⊃ ~ Hba	5 ∀E
7	~ Hba	4, 6 ⊃E
8	(∃y) ~ Hya	7 ∃I
9	(∃y) ~ Hya	3, 4–9 ∃E
10	(∀x)(∃y) ~ Hyx	9 ∀I

We have met all the restrictions for using each of the two rules Existential Elimination and Universal Introduction. The constant we had to worry about in using Existential Elimination is 'b', for it is the instantiating constant used to form a substitution instance of '(∃y)Hay' at line 4. By choosing 'b' as the instantiating constant we were able to meet all the restrictions on Existential Elimination: 'b' does not occur in any assumption that is open at line 9, does not occur in the existentially quantified sentence '(∃y)Hay' at line 3, and does not occur in the sentence '(∃y) ~ Hya' derived by Existential Elimination at line 9.

Our next argument is somewhat more complex, having one premise that contains three quantifiers:

(∀x)[(∃z)Fxz ⊃ (∀y)Fxy]
(∃x)(∃y)Fxy
———————————
(∃x)(∀w)Fxw

The argument is valid in *PD,* and the derivation is not as difficult as may be feared. We will take our first clue from the second assumption, which begins with two existential quantifiers. This suggests we will be using Existential Elimination twice, as follows:

Derive: (∃x)(∀y)Fxy

1	(∀x)[(∃z)Fxz ⊃ (∀y)Fxy]	Assumption
2	(∃x)(∃y)Fxy	Assumption
3	(∃y)Fay	A / ∃E
4	Fab	A / ∃E
G	(∃x)(∀w)Fxw	∃I
G	(∃x)(∀w)Fxw	3, 4–__ ∃E
G	(∃x)(∀w)Fxw	2, 3–__ ∃E

We next use Universal Elimination to produce a conditional to which we can apply Conditional Elimination after applying Existential Introduction to the assumption on line 4, being careful to choose an instantiating constant that will produce a match between the conditional and the existentially quantified sentence we generate. Here the instantiating constant ‘a’ does the trick:

Derive: $(\exists x)(\forall y)Fxy$

1	$(\forall x)[(\exists z)Fxz \supset (\forall y)Fxy]$	Assumption
2	$(\exists x)(\exists y)Fxy$	Assumption
3	$\quad (\exists y)Fay$	A / ∃E
4	$\quad\quad Fab$	A / ∃E
5	$\quad\quad (\exists z)Faz \supset (\forall y)Fay$	1 ∀E
6	$\quad\quad (\exists z)Faz$	4 ∃I
7	$\quad\quad (\forall y)Fay$	5, 6 ⊃E
G	$\quad\quad (\exists x)(\forall w)Fxw$	∃I
G	$\quad (\exists x)(\forall w)Fxw$	3, 4–__ ∃E
G	$(\exists x)(\forall w)Fxw$	2, 3–__ ∃E

Our current goal is ‘$(\exists x)(\forall w)Fxw$’. To obtain it, by Existential Introduction, we need to first derive a substitution instance of that sentence, say ‘$(\forall w)Faw$’. We have already derived ‘$(\forall y)Fay$’. This is not the sentence we need, because it contains the variable ‘y’ where we want ‘w’. But we can easily obtain the substitution instance we want by using Universal Elimination (with a new instantiating constant) followed by Universal Introduction using the variable ‘y’ instead of the variable ‘w’. We do this at lines 8 and 9, completing the derivation:

Derive: $(\exists x)(\forall y)Fxy$

1	$(\forall x)[(\exists z)Fxz \supset (\forall y)Fxy]$	Assumption
2	$(\exists x)(\exists y)Fxy$	Assumption
3	$\quad (\exists y)Fay$	A / ∃E
4	$\quad\quad Fab$	A / ∃E
5	$\quad\quad (\exists z)Faz \supset (\forall y)Fay$	1 ∀E
6	$\quad\quad (\exists z)Faz$	4 ∃I
7	$\quad\quad (\forall y)Fay$	5, 6 ⊃E
8	$\quad\quad Fac$	7 ∀E
9	$\quad\quad (\forall w)Faw$	8 ∀I
10	$\quad\quad (\exists x)(\forall w)Fxw$	9 ∃I
11	$\quad (\exists x)(\forall w)Fxw$	3, 4–10 ∃E
12	$(\exists x)(\forall w)Fxw$	2, 3–11 ∃E

As a final example consider the following argument:

Everyone loves a lover.

Tom loves Alice.

Everyone loves everyone.

Assuming the predicate 'loves' is being used unambiguously, this argument is, perhaps surprisingly, valid. We can reason informally as follows: Because Tom loves Alice, Tom is a lover. And since everyone loves a lover, everyone loves Tom. But then everyone is a lover, and since everyone loves a lover, everyone loves everyone. Here is a symbolization of the argument in *PL:*

$(\forall x)[(\exists y)Lxy \supset (\forall z)Lzx]$

Lta

$(\forall x)(\forall y)Lxy$

As in the last example, it appears that our ultimate goal will be obtained by Universal Introduction, and indeed that our penultimate goal will also be obtained by this rule. Our work would be over if we could proceed as follows:

Derive: $(\forall x)(\forall y)Lxy$

1	$(\forall x)[(\exists y)Lxy \supset (\forall z)Lzx]$	Assumption	
2	Lta	Assumption	
3	$(\forall y)Lty$	2 ∀I	**MISTAKE!**
4	$(\forall x)(\forall y)Lxy$	3 ∀I	**MISTAKE!**

But of course we cannot do this. Both line 3 and line 4 are in violation of the restrictions on Universal Introduction. In each case the constant we are replacing, first 'a' and then 't', occurs in an open assumption (at line 2). To use Universal Introduction we need to obtain a sentence like 'Lta' but formed from other constants, any other constants. We select 'c' and 'd':

Derive: $(\forall x)(\forall y)Lxy$

1	$(\forall x)[(\exists y)Lxy \supset (\forall z)Lzx]$	Assumption
2	Lta	Assumption
G	Lcd	
G	$(\forall y)Lcy$	__ ∀I
G	$(\forall x)(\forall y)Lxy$	__ ∀I

How might we obtain our current goal, 'Lcd'? Recall the reasoning we did in English: from Lta we can infer that Tom is a lover—and we mirror this inference in *PD* by obtaining '(∃y)Lty' by Existential Introduction. In English we reasoned that if Tom is a lover, then everyone loves Tom. We can mirror this in *PD* by applying Universal Elimination to line 1. And since we have established that Tom is a lover, we can infer that everyone loves him. So we have:

Derive: (∀x)(∀y)Lxy

1	(∀x)[(∃y)Lxy ⊃ (∀z)Lzx]	Assumption
2	Lta	Assumption
3	(∃y)Lty	2 ∃I
4	(∃y)Lty ⊃ (∀z)Lzt	1 ∀E
5	(∀z)Lzt	3, 4 ⊃E
G	Lcd	
G	(∀y)Lcy	__ ∀I
G	(∀x)(∀y)Lxy	__ ∀I

It is because neither 'c' nor 'd' occur in an open assumption that we will be able to derive our final goal by two uses of Universal Introduction. But how do we get, in *PD,* from line 5 to 'Lcd'? From line 5 we can get 'Ldt' by Universal Elimination. But how does this help us get 'Lcd'? One difference between these two sentences is that 'd' occurs in the first position after 'L' in the first, and in the second position in the second. We also note that line 1, which is our symbolization of 'Everyone loves a lover' contains two occurrences of the two-place predicate 'L', with 'x' occurring in the first position after L in the first occurrence, and in the second position in the second occurrence. So perhaps we can use this sentence to move 'd' from the first position after L to the second position. (Remember that 'Everyone loves a lover' does say that if someone loves then that person gets loved.) Following this clue we proceed as follows:

Derive: (∀x)(∀y)Lxy

1	(∀x)[(∃y)Lxy ⊃ (∀z)Lzx]	Assumption
2	Lta	Assumption
3	(∃y)Lty	2 ∃I
4	(∃y)Lty ⊃ (∀z)Lzt	1 ∀E
5	(∀z)Lzt	3, 4 ⊃E
6	Ldt	5 ∀E
7	(∃y)Ldy	6 ∃I
8	(∃y)Ldy ⊃ (∀z)Lzd	1 ∀E
9	(∀z)Lzd	7, 8 ⊃E
10	Lcd	9 ∀E
11	(∀y)Lcy	10 ∀I
12	(∀x)(∀y)Lxy	11 ∀I

Our derivation is now complete.

'(∀z)[Fz ⊃ (Fz ∨ Gz)]' is a theorem in *PD*. To prove that it is such we need to derive it from the empty set, which means we will need a derivation that has no primary assumptions. The most plausible strategy for obtaining this sentence is Universal Introduction.

```
Derive: (∀z)[Fz ⊃ (Fz ∨ Gz)]

  |
  |
G | Fb ⊃ (Fb ∨ Gb)
G | (∀z)[Fz ⊃ (Fz ∨ Gz)]            __ ∀I
```

Our current goal is a material conditional and can be obtained by Conditional Introduction, using Disjunction Introduction to derive 'Fb ∨ Gb' within the Conditional Introduction subderivation:

```
Derive: (∀z)[Fz ⊃ (Fz ∨ Gz)]

1 | | Fb                          A / ⊃I
  | |----
2 | | Fb ∨ Gb                     2 ∨I
3 | Fb ⊃ (Fb ∨ Gb)                1–2 ⊃I
4 | (∀z)[Fz ⊃ (Fz ∨ Gz)]          4 ∀I
```

We have met both of the restrictions on Universal Introduction. The instantiating constant 'b' does not occur in any assumption that is open at line 4 and does not occur in the sentence derived on line 4 by Universal Introduction.

To prove the theorem '(∃x)Fx ⊃ (∃x)(Fx ∨ Gx)' we will use Conditional Introduction, Existential Elimination, and Existential Introduction as well as Disjunction Introduction. The proof is straightforward:

```
Derive: (∃x)Fx ⊃ (∃x)(Fx ∨ Gx)

1 | | (∃x)Fx                      A / ⊃I
  | |--------
2 | | | Fa                        A / ∃E
  | | |----
3 | | | Fa ∨ Ga                   2 ∨I
4 | | | (∃x)(Fx ∨ Gx)             3 ∃I
5 | | (∃x)(Fx ∨ Gx)               1, 2–4 ∃E
6 | (∃x)Fx ⊃ (∃x)(Fx ∨ Gx)        1–5 ⊃I
```

We used Conditional Introduction as our primary strategy because our ultimate goal is a material conditional. We used Existential Elimination within that strategy because the assumption that begins the Conditional Introduction subderivation is an existentially quantified sentence. And we used Existential Introduction at line 4, within our Existential Elimination subderivation, to generate the consequent of the goal conditional. The consequent does not contain the instantiating constant 'a' and can therefore be pulled out of the Existential Elimination subderivation.

The third theorem we will prove is '$(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$'. This is also a material conditional, and our primary strategy will again be Conditional Introduction. The assumption of our Conditional Introduction subderivation will be an existentially quantified sentence, suggesting that we use Existential Elimination within our Conditional Introduction subderivation. And if we can derive '$(\exists x)(\exists y)Fxy$' within our Existential Elimination subderivation we will be able to end that subderivation and complete our derivation:

```
Derive: (∃x)(∀y)Fxy ⊃ (∃x)(∃y)Fxy

1 |  | (∃x)(∀y)Fxy                       Assumption
  |  |-------------
2 |  |  | (∀y)Fay                         A / ∃E
  |  |  |---------
  |  |  |
  |  |  |
G |  |  | (∃x)(∃y)Fxy
G |  | (∃x)(∃y)Fxy                        1, 2–__ ∃E
G | (∃x)(∀y)Fxy ⊃ (∃x)(∃y)Fxy             1–__ ⊃I
```

Completing this derivation is now straightforward. We apply Universal Elimination to the sentence on line 2 to produce 'Fab' and then use Existential Introduction twice to derive '$(\exists x)(\exists y)Fxy$'.

```
Derive: (∃x)(∀y)Fxy ⊃ (∃x)(∃y)Fxy

1 |  | (∃x)(∀y)Fxy                       Assumption
  |  |-------------
2 |  |  | (∀y)Fay                         A / ∃E
  |  |  |---------
3 |  |  | Fab                             2 ∀E
4 |  |  | (∃y)Fay                         3 ∃I
5 |  |  | (∃x)(∃y)Fxy                     4 ∃I
6 |  | (∃x)(∃y)Fxy                        1, 2–5 ∃E
7 | (∃x)(∀y)Fxy ⊃ (∃x)(∃y)Fxy             1–6 ⊃I
```

We have met all the restrictions on Existential Elimination. The instantiating constant 'a' does not occur in any assumption that is open as of line 6. The constant 'a' also does not occur in the existentially quantified sentence to which we are applying Existential Elimination, and it does not occur in the sentence derived at line 6 by Existential Elimination.

It is worth noting that since there are no restrictions on Existential Introduction, we could have entered 'Faa' rather than 'Fab' at line 3 (there are also no restrictions on Universal Elimination), and then applied Existential Introduction twice.

The last theorem we will consider is the quantified sentence '$(\exists x)(Fx \supset (\forall y)Fy)$'. At first glance it appears that we should use Existential Introduction to derive this sentence from some substitution instance, for example, ‘Fa ⊃ (∀y)Fy’ and so the latter sentence should be a subgoal. However, this will not work! ‘Fa ⊃ (∀y)Fy’ is not quantificationally true and therefore cannot be derived in *PD* from no assumptions. So we must choose another strategy. Our primary strategy will be Negation Elimination and the proof will be quite complicated:

```
Derive: (∃x)(Fx ⊃ (∀y)Fy)

1  |  | ~ (∃x)(Fx ⊃(∀y)Fy)          A / ~ E
   |  |---------------------
   |  |
   |  |
G  |  | (∃x)(Fx ⊃ (∀y)Fy)
   |  | ~ (∃x)(Fx ⊃ (∀y)Fy)         1 R
G  | (∃x)(Fx ⊃ (∀y)Fy)             1–__ ~ E
```

We have selected Negation Elimination as our primary strategy because there is no plausible alternative to that strategy. We have selected ‘(∃x)(Fx ⊃ (∀y)Fy)’ and ‘~ (∃x)(Fx ⊃ (∀y)Fy)’ as the contradictory sentences we will derive within that strategy because the latter sentence is our assumption on line 1 and therefore available for use. The question now is how to derive ‘(∃x)(Fx ⊃ (∀y)Fy)’. Since this is an existentially quantified sentence we will attempt to derive it by Existential Introduction: first deriving the substitution instance ‘Fa ⊃ (∀y)Fy’ of that sentence (any other instantiating constant could be used). The substitution instance should be derivable using Conditional Introduction:

```
Derive: (∃x)(Fx ⊃ (∀y)Fy)

1  |  | ~ (∃x)(Fx ⊃ (∀y)Fy)         A / ~ E
   |  |---------------------
2  |  |  | Fa                       A / ⊃I
   |  |  |---
   |  |  |
G  |  |  | (∀y)Fy
G  |  | Fa ⊃ (∀y)Fy                 2–__ ⊃I
G  |  | (∃x)(Fx ⊃ (∀y)Fy)           __ ∃I
   |  | ~ (∃x)(Fx ⊃ (∀y)Fy)         1 R
G  | (∃x)(Fx ⊃ (∀y)Fy)             1–__ ~ E
```

Our new goal is ‘(∀y)Fy’, a universally quantified sentence. We cannot obtain it by applying Universal Introduction to the sentence on line 2, because ‘a’ here occurs in an open assumption. So we will try to obtain a different substitution instance of '(∀y)Fy', 'Fb', and we will try to derive this substitution instance using Negation Elimination:

```
1 |  | ~ (∃x)(Fx ⊃ (∀y)Fy)          A / ~ E
  |  |-------------------
2 |  |  | Fa                         A / ⊃I
  |  |  |---
3 |  |  |  | ~ Fb                    A / ~ E
  |  |  |  |-----
  |  |  |  |
  |  |  |  |
G |  |  | Fb
G |  |  | (∀y)Fy                     ___ ∀I
G |  | Fa ⊃ (∀y)Fy                   2–___ ⊃I
G |  | (∃x)(Fx ⊃ (∀y)Fy)
G |  | ~ (∃x)(Fx ⊃ (∀y)Fy)           1 R
G | (∃x)(Fx ⊃ (∀y)Fy)               1–___ ~ E
```

We now have to decide on the sentence and its negation to be derived within the Negation Elimination subderivation. Two negations are accessible at this point: '~ Fb' and '~ (∃x)(Fx ⊃ (∀y)Fy)'. We will make the latter sentence and '(∃x)(Fx ⊃ (∀y)Fy)' our goals as picking 'Fb' and '~ Fb' as goals appears to be unpromising (there is no obvious way to derive 'Fb' from the assumptions on lines 1–3). We plan to derive '(∃x)(Fx ⊃ (∀y)Fy)' using Existential Introduction:

```
1 |  | ~ (∃x)(Fx ⊃ (∀y)Fy)          A / ~ E
  |  |-------------------
2 |  |  | Fa                         A / ⊃I
  |  |  |---
3 |  |  |  | ~ Fb                    A / ~ E
  |  |  |  |-----
  |  |  |  |
  |  |  |  |
G |  |  |  | Fb ⊃ (∀y)Fy
G |  |  |  | (∃x)(Fx ⊃ (∀y)Fy)      ___ ∃I
  |  |  |  | ~ (∃x)(Fx ⊃ (∀y)Fy)    1 R
G |  |  | Fb
G |  |  | (∀y)Fy                     ___ ∀I
G |  | Fa ⊃ (∀y)Fy                   2–___ ⊃I
G |  | (∃x)(Fx ⊃ (∀y)Fy)
  |  | ~ (∃x)(Fx ⊃ (∀y)Fy)           1 R
G | (∃x)(Fx ⊃ (∀y)Fy)               1–___ ~ E
```

We have selected 'b' as the instantiating constant in our new goal because we anticipate using Conditional Introduction to derive 'Fb ⊃ (∀y)Fy', and this use of 'b' will give us 'Fb' as an assumption, something that is likely to be useful as we already have '~ Fb' at line 3.

1	| | ~ (∃x)(Fx ⊃ (∀y)Fy)	A / ~ E
2	| | | Fa	A / ⊃I
3	| | | | ~ Fb	A / ~ E
4	| | | | | Fb	A / ⊃I
G	| | | | | (∀y)Fy	
G	| | | | Fb ⊃ (∀y)Fy	
G	| | | | (∃x)(Fx ⊃ (∀y)Fy)	___ ∃I
	| | | | ~ (∃x)(Fx ⊃ (∀y)Fy)	1 R
G	| | | Fb	
G	| | | (∀y)Fy	___ ∀I
G	| | Fa ⊃ (∀y)Fy	2–___ ⊃I
G	| | (∃x)(Fx ⊃ (∀y)Fy)	
	| | ~ (∃x)(Fx ⊃ (∀y)Fy)	1 R
G	| (∃x)(Fx ⊃ (∀y)Fy)	1–___ ~ E

Our new goal is '(∀y)Fy' and since 'Fb' and '~ Fb' are both accessible, we can easily derive it using Negation Elimination, completing the derivation:

Derive: (∃x)(Fx ⊃ (∀y)Fy)

1	| | ~ (∃x)(Fx ⊃ (∀y)Fy)	A / ~ E
2	| | | Fa	A / ⊃I
3	| | | | ~ Fb	A / ~ I
4	| | | | | Fb	A / ⊃I
5	| | | | | | ~ (∀y)Fy	A / ~ E
6	| | | | | | Fb	4 R
7	| | | | | | ~ Fb	3 R
8	| | | | | (∀y)Fy	5–7 ~ E
9	| | | | Fb ⊃ (∀y)Fy	4–8 ⊃I
10	| | | | (∃x)(Fx ⊃ (∀y)Fy)	9 ∃I
11	| | | | ~ (∃x)(Fx ⊃ (∀y)Fy)	1 R
12	| | | Fb	3–11 ~ E
13	| | | (∀y)Fy	12 ∀I
14	| | Fa ⊃ (∀y)Fy	2–13 ⊃I
15	| | (∃x)(Fx ⊃ (∀y)Fy)	14 ∃I
16	| | ~ (∃x)(Fx ⊃ (∀y)Fy)	1 R
17	| (∃x)(Fx ⊃ (∀y)Fy)	1–16 ~ E

This is a complex derivation, as we warned it would be. In the end we used the same pair of contradictory sentences in two Negation Elimination subderivations. This sometimes happens.

EQUIVALENCE

To show that sentences **P** and **Q** of *PL* are equivalent in *PD* we must derive each from the unit set of the other. As our first example we take the sentences '(∀x)(Fa ⊃ Fx)' and 'Fa ⊃ (∀x)Fx'. We begin by deriving the second of these sentences from the first, and since our goal sentence in this derivation is a material conditional, we will use Conditional Introduction:

Derive: Fa ⊃ (∀x)Fx

1	(∀x)(Fa ⊃ Fx)	Assumption
2	Fa	A / ⊃I
G	(∀x)Fx	
G	Fa ⊃ (∀x)Fx	2–__ ⊃I

We cannot derive our present goal, '(∀x)Fx', by simply applying Universal Introduction to 'Fa' at line 2, for the sentence on line 2 is an open assumption and 'a' occurs in that sentence. We can rather try to derive a different substitution instance of '(∀x)Fx', say 'Fb', and then apply Universal Introduction. And this is easy to do by applying Universal Elimination to the sentence on line 1 (being careful to use an instantiating constant other than 'a'), and then using Conditional Elimination:

Derive: Fa ⊃ (∀x)Fx

1	(∀x)(Fa ⊃ Fx)	Assumption
2	Fa	A / ⊃I
3	Fa ⊃ Fb	1 ∀E
4	Fb	2, 3 ⊃E
5	(∀x)Fx	4 ∀I
6	Fa ⊃ (∀x)Fx	2–5 ⊃I

We have met both restrictions on Universal Introduction at line 5: the instantiating constant 'b' does not occur in any open assumption; nor does it occur in the derived sentence '(∀x)Fx'.

We must now derive '(∀x)(Fa ⊃ Fx)' from 'Fa ⊃ (∀x)Fx'. A plausible start is

Derive: (∀x)(Fa ⊃ Fx)

1	Fa ⊃ (∀x)Fx	Assumption
2	Fa	A / ⊃I
G	Fb	
G	Fa ⊃ Fb	2–__ ⊃I
G	(∀x)(Fa ⊃ Fx)	__ ∀I

We plan to derive the last sentence by Universal Introduction, and the substitution instance on the prior line by Conditional Introduction. And we can now see how to complete the derivation. We can apply Conditional Elimination to the sentences on lines 1 and 2 to derive '(∀x)Fx', from which we can then derive 'Fb':

```
Derive: (∀x)(Fa ⊃ Fx)

1 | Fa ⊃ (∀x)Fx              Assumption
  |------------
2 | | Fa                     A / ⊃I
  | |---
3 | | (∀x)Fx                 1, 2 ⊃E
4 | | Fb                     3 ∀E
5 | Fa ⊃ Fb                  2–4 ⊃I
6 | (∀x)(Fa ⊃ Fx)            5 ∀I
```

Having derived each member of our pair of sentences from the other, we have demonstrated that the sentences '(∀x)(Fa ⊃ Fx)' and 'Fa ⊃ (∀x)Fx' are equivalent in *PD*.

We will next show that '(∀x)Fx ⊃ Ga' and '(∃x)(Fx ⊃ Ga)' are equivalent in *PD*. It is reasonably straightforward to derive '(∀x)Fx ⊃ Ga' from '(∃x)(Fx ⊃ Ga)'. We begin with

```
Derive: (∀x)Fx ⊃ Ga

1 | (∃x)(Fx ⊃ Ga)            Assumption
  |--------------
2 | | (∀x)Fx                 A / ⊃I
  | |-------
  | |
  | |
  | |
G | | Ga
G | (∀x)Fx ⊃ Ga              2–__ ⊃I
```

We will complete the derivation by using Existential Elimination—being careful to use an instantiating constant other than 'a' (because 'a' occurs in 'Ga', the sentence we plan to derive with Existential Elimination):

```
Derive: (∀x)Fx ⊃ Ga

1 | (∃x)(Fx ⊃ Ga)            Assumption
  |--------------
2 | | (∀x)Fx                 A / ⊃I
  | |-------
3 | | | Fb ⊃ Ga            A / ∃E
  | | |--------
4 | | | Fb                 2 ∀E
5 | | | Ga                 3–4 ⊃E
6 | | Ga                     1, 3–5 ∃E
7 | (∀x)Fx ⊃ Ga              2–6 ⊃I
```

Our use of Existential Elimination at line 6 meets all three restrictions on that rule: the instantiating constant ‘b’ does not occur in ‘(∃x) (Fx ⊃ Ga)’, does not occur in any assumption that is open at line 6, and does not occur in the sentence ‘Ga’ that we derived with Existential Elimination.

Deriving ‘(∃x) (Fx ⊃ Ga)’ from ‘(∀x)Fx ⊃ Ga’ is a somewhat more challenging exercise. Since our primary goal is an existentially quantified sentence, both Existential Introduction and Negation Elimination suggest themselves as primary strategies. We have opted to use Negation Elimination, and since the assumption that begins that strategy is a negation, we will make it and the sentence of which it is a negation our goals within the Negation Elimination subderivation:

```
Derive: (∃x) (Fx ⊃ Ga)

1 | (∀x)Fx ⊃ Ga                  Assumption
  |-------------
2 |  | ~ (∃x) (Fx ⊃ Ga)           A / ~ E
  |  |-----------------
  |  |
  |  |
G |  |  (∃x) (Fx ⊃ Ga)
  |  | ~ (∃x) (Fx ⊃ Ga)           2 R
G | (∃x) (Fx ⊃ Ga)                1–__ ~ E
```

When two primary strategies suggest themselves, it is frequently useful to use one as a secondary strategy within the other, primary strategy. Here we will use Existential Introduction as a secondary strategy: We will try to obtain the goal ‘(∃x) (Fx ⊃ Ga)’ by Existential Introduction, first using Conditional Introduction to derive an appropriate substitution instance of the goal sentence:

```
Derive: (∃x) (Fx ⊃ (∀y)Fy)

1 | (∀x)Fx ⊃ Ga                  Assumption
  |-------------
2 |  | ~ (∃x) (Fx ⊃ Ga)           A / ~ E
  |  |-----------------
3 |  |  | Fa                      A / ⊃I
  |  |  |---
  |  |  |
  |  |  |
G |  |  | Ga
G |  |  Fa ⊃ Ga                   3–__ ⊃I
G |  |  (∃x) (Fx ⊃ Ga)            __ ∃I
  |  | ~ (∃x) (Fx ⊃ Ga)           1 R
G | (∃x) (Fx ⊃ Ga)                1–__ ~ E
```

The current goal, 'Ga', can be derived by Conditional Elimination using the sentence on line 1 *if* we can first derive the antecedent '(∀x)Fx' of that sentence. It is not easy to see how the antecedent might be derived, but one strategy is to try to first derive a substitution instance in which the instantiating constant does not occur in an open assumption. This rules out 'Fa'. So we will try to derive 'Fb', and since no more direct strategy suggests itself at this point, we'll try to derive 'Fb' by Negation Elimination:

Derive: (∃x) (Fx ⊃ (∀y)Fy)

1	| (∀x)Fx ⊃ Ga	Assumption
2	| | ~ (∃x) (Fx ⊃ Ga)	A / ~ E
3	| | | Fa	A / ⊃I
4	| | | | ~ Fb	A / ~ E
G	| | | Fb	4–__ ~ E
G	| | | (∀x)Fx	__ ∀I
G	| | | Ga	1, __ ⊃E
G	| | Fa ⊃ Ga	3–__ ⊃I
G	| | (∃x) (Fx ⊃ Ga)	∃I
	| | ~ (∃x) (Fx ⊃ Ga)	1 R
G	| (∃x) (Fx ⊃ Ga)	1–__ ~ E

Given '~ Fb' at line 4 we can obtain 'Fb ⊃ Ga'. We know we can do this because we know that given the negation of the antecedent of any conditional we can derive the conditional—as the following schema demonstrates:

n	| ~ **P**	
n+1	| | **P**	A / ⊃I
n+2	| | | ~ **Q**	A / ~ E
n+3	| | | **P**	n+1 R
n+4	| | | ~ **P**	n R
n+5	| | **Q**	n+2–n+4 ~ E
n+6	| **P** ⊃ **Q**	n+1–n+5 ⊃I

Once we derive 'Fb ⊃ Ga' we can obtain '(∃x) (Fx ⊃ Ga)' by Existential Introduction. Because we already have the negation of that sentence at line 2 we can see our way clear to deriving a sentence and its negation as follows:

Derive: (∃x)(Fx ⊃ Ga)

1	(∀x)Fx ⊃ Ga	Assumption
2	~ (∃x)(Fx ⊃ Ga)	A / ~ E
3	Fa	A / ⊃I
4	~ Fb	A / ~ E
5	Fb	A / ⊃I
6	~ Ga	A / ~ E
7	Fb	5 R
8	~ Fb	4 R
9	Ga	6–8 ~ E
10	Fb ⊃ Ga	5–9 ⊃I
11	(∃x)(Fx ⊃ Ga)	10 ∃I
12	~ (∃x)(Fx ⊃ Ga)	2 R
13	Fb	4–12 ~ E
14	(∀x)Fx	13 ∀I
15	Ga	1, 14 ⊃E
16	Fa ⊃ Ga	3–15 ⊃I
17	(∃x)(Fx ⊃ Ga)	16 ∃I
18	~ (∃x)(Fx ⊃ Ga)	2 R
19	(∃x)(Fx ⊃ Ga)	1–18 ~ E

We will conclude our discussion of Equivalence in *PD* by deriving each of the following sentences from the unit set of the other:

(∀x)[Fx ⊃ (∃y)Gxy]   (∀x)(∃y)(Fx ⊃ Gxy)

Establishing that these sentences are equivalent in *PD* is substantially more difficult than was establishing equivalence in our last example, in large part because in these sentences the existentially quantified formulas occur *within the scope of* universal quantifiers. We begin by deriving '(∀x)(∃y)(Fx ⊃ Gxy)' from {(∀x)[Fx ⊃ (∃y)Gxy]}. Since our one primary assumption will be a universally quantified sentence, as will our goal, it is plausible to expect that we will use both Universal Elimination and Universal Introduction:

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	Fa ⊃ (∃y)Gay	1 ∀E
G	(∃y)(Fa ⊃ Gay)	
G	(∀x)(∃y)(Fx ⊃ Gxy)	__ ∀I

It is now tempting to make ‘Fa ⊃ Gab’ our next subgoal, to be derived using Conditional Introduction. And if we can obtain ‘Fa ⊃ Gab’ we can go on to derive ‘(∃y)(Fa ⊃ Gay)’ by Existential Introduction:

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	Fa ⊃ (∃y)Gay	1 ∀E
3	Fa	A / ⊃I
G	Gab	
G	Fa ⊃ Gab	3–___ ⊃I
G	(∃y)(Fa ⊃ Gay)	___ ∃I
G	(∀x)(∃y)(Fx ⊃ Gxy)	___ ∀I

‘(∃y)Gay’ can be derived from lines 2 and 3 by Conditional Elimination. We might then plan to use Existential Elimination to get from ‘(∃y)Gay’ to the current goal sentence ‘Gab’. But we have to be careful here. If we want to derive ‘Gab’ by Existential Elimination then the instantiating constant for Existential Elimination has to be a constant other than ‘b’.

Derive: (∀x)(∃y)(Fx ⊃ Gxy)

1	(∀x)[Fx ⊃ (∃y)Gxy]	Assumption
2	Fa ⊃ (∃y)Gay	1 ∀E
3	Fa	A / ⊃I
4	(∃y)Gay	2, 3 ⊃E
5	Gac	A / ∃E
G	Gab	
G	Gab	4, 5–___ ∃E
G	Fa ⊃ Gab	3–___ ⊃I
G	(∃y)(Fa ⊃ Gay)	___ ∃I
G	(∀x)(∃y)(Fx ⊃ Gxy)	___ ∀I

But how do we get from ‘Gac’ to ‘Gab’? A negation strategy might work, but it would be complicated as there are no negations among the accessible sentences.

It is time to consider an alternative strategy. We will try to obtain our penultimate goal, ‘(∃y) (Fa ⊃ Gay)’, by Negation Elimination rather than by Existential Introduction:

```
Derive: (∀x) (∃y) (Fx ⊃ Gxy)

1 | (∀x) [Fx ⊃ (∃y) Gxy]          Assumption
  |---------------------
2 |  | ~ (∃y) (Fa ⊃ Gay)          A / ~ E
  |  |------------------
  |  |
  |  |
G |  |   (∃y) (Fa ⊃ Gay)          15 ∃I
G |  | ~ (∃y) (Fa ⊃ Gay)          2 R
G | (∃y) (Fa ⊃ Gay)               2–17 ~ E
G | (∀x) (∃y) (Fx ⊃ Gxy)          18 ∀I
```

It may appear that because ‘(∃y) (Fa ⊃ Gay)’ is still our goal we are making no progress. But this is not so, for we now have an additional assumption to work from. We will now proceed much as we did in our first attempt at this derivation:

```
Derive: (∀x) (∃y) (Fx ⊃ Gxy)

1 | (∀x) [Fx ⊃ (∃y) Gxy]          Assumption
  |---------------------
2 |  | ~ (∃y) (Fa ⊃ Gay)          A / ~ E
  |  |------------------
3 |  |  | Fa                      A / ⊃I
  |  |  |---
4 |  |  | Fa ⊃ (∃y) Gay           1 ∀E
5 |  |  | (∃y) Gay                3, 4 ⊃E
6 |  |  |  | Gac                  A / ∃E
  |  |  |  |-----
  |  |  |  |
G |  |  |  | Gab
G |  |  | Gab                     5, 6–__ ∃E
G |  | Fa ⊃ Gab                   3–__ ⊃I
G |  | (∃y) (Fa ⊃ Gay)            __ ∃I
G |  | ~ (∃y) (Fa ⊃ Gay)          2 R
G | (∃y) (Fa ⊃ Gay)               2–__ ~ E
G | (∀x) (∃y) (Fx ⊃ Gxy)          __ ∀I
```

Once again we want to get from ‘Gac’ to ‘Gab’. But this time we do have an accessible negation, ‘~ (∃y) (Fa ⊃ Gay)’. So we will use a negation strategy, assuming ‘~ Gab’ and seeking to derive ‘(∃y) (Fa ⊃ Gay)’ along with reiterating its negation:

Derive: (∀x) (∃y) (Fx ⊃ Gxy)

```
 1 | (∀x) [Fx ⊃ (∃y) Gxy]                 Assumption
   |---------------------
 2 | | ~ (∃y) (Fa ⊃ Gay)                  A / ~ E
   | |-------------------
 3 | | | Fa                               A ⊃I
   | | |----
 4 | | | Fa ⊃ (∃y) Gay                    1 ∀E
 5 | | | (∃y) Gay                         3, 4 ⊃E
 6 | | | | Gac                            A / ∃E
   | | | |-----
 7 | | | | | ~ Gab                        A ~ E
   | | | | |-------
   | | | | |
 G | | | | | (∃y) (Fa ⊃ Gay)
   | | | | | ~ (∃y) (Fa ⊃ Gay)            2 R
 G | | | | Gab                            7–__ ~ E
 G | | | Gab                              5, 6–__ ∃E
 G | | Fa ⊃ Gab                           3–__ ⊃I
 G | | (∃y) (Fa ⊃ Gay)                    __ ∃I
   | | ~ (∃y) (Fa ⊃ Gay)                  2 R
 G | (∃y) (Fa ⊃ Gay)                      2–__ ~ E
 G | (∀x) (∃y) (Fx ⊃ Gxy)                 __ ∀I
```

What remains is to derive '(∃y) (Fa ⊃ Gay)'. This is easily done. We assume 'Fa', derive 'Gac' by Reiteration, derive 'Fa ⊃ Gac' by Conditional Introduction, and then '(∃y) (Fa ⊃ Gay)' by Existential Introduction. The derivation is then complete:

Derive: (∀x) (∃y) (Fx ⊃ Gxy)

```
 1 | (∀x) [Fx ⊃ (∃y) Gxy]                 Assumption
   |---------------------
 2 | | ~ (∃y) (Fa ⊃ Gay)                  A / ~ E
   | |-------------------
 3 | | | Fa                               A ⊃I
   | | |----
 4 | | | Fa ⊃ (∃y) Gay                    1 ∀E
 5 | | | (∃y) Gay                         3, 4 ⊃E
 6 | | | | Gac                            A / ∃E
   | | | |-----
 7 | | | | | ~ Gab                        A ~ E
   | | | | |-------
 8 | | | | | | Fa                         A / ⊃I
   | | | | | |----
 9 | | | | | | Gac                        6 R
10 | | | | | Fa ⊃ Gac                     8–9 ⊃I
11 | | | | | (∃y) (Fa ⊃ Gay)              10 ∃I
12 | | | | | ~ (∃y) (Fa ⊃ Gay)            2 R
13 | | | | Gab                            7–12 ~ E
14 | | | Gab                              5, 6–13 ∃E
15 | | Fa ⊃ Gab                           3–14 ⊃I
16 | | (∃y) (Fa ⊃ Gay)                    15 ∃I
17 | | ~ (∃y) (Fa ⊃ Gay)                  2 R
18 | (∃y) (Fa ⊃ Gay)                      2–17 ~ E
19 | (∀x) (∃y) (Fx ⊃ Gxy)                 18 ∀I
```

We must now derive '(∀x) [Fx ⊃ (∃y)Gxy]' from '(∀x) (∃y) (Fx ⊃ Gxy)'. This will be an easier task since we can derive '(∃y) (Fa ⊃ Gay)' by Universal Elimination and then do the bulk of the derivation within an Existential Elimination subderivation:

Derive: (∀x) [Fx ⊃ (∃y)Gxy]

1	(∀x) (∃y) (Fx ⊃ Gxy)	Assumption
2	(∃y) (Fa ⊃ Gay)	1 ∀E
3	Fa ⊃ Gab	A / ∃E
4	Fa	A ⊃I
5	Gab	3, 4 ⊃E
6	(∃y)Gay	5 ∃I
7	Fa ⊃ (∃y)Gay	4–6 ⊃I
8	Fa ⊃ (∃y)Gay	3, 4–7 ∃E
9	(∀x) [Fx ⊃ (∃y)Gxy]	8 ∀I

The instantiating constant 'b' for our use of Existential Elimination does not occur in the existentially quantified sentence '(∃y) (Fa ⊃ Gay)', in any assumption that is open at line 8, or in the sentence 'Fa ⊃ (∃y)Gay' obtained by Existential Elimination. (In this case we could also have applied Universal Introduction *within* the Existential Elimination subderivation and then moved '(∀x) [Fx ⊃ (∃y)Gxy]' out of that subderivation.) This completes our demonstration that '(∀x) [Fx ⊃ (∃y)Gxy]' and '(∀x) (∃y) (Fx ⊃ Gxy)' are equivalent in *PD*.

INCONSISTENCY

We next turn our attention to demonstrating that sets of sentences of *PL* are inconsistent in *PD*. Recall that a set of sentences is inconsistent in *PD* if we can derive both a sentence **Q** and its negation ~ **Q** from the set. As our first example we will show that the set {(∀x) (Fx ≡ Gx), (∃y) (Fy & ~ Gy)} is inconsistent in *PD*. Because this set does not contain a negation, it is not obvious what our **Q** and ~ **Q** should be. We will use the set member '(∀x) (Fx ≡ Gx)' as **Q**, making ~ **Q** '~ (∀x) (Fx ≡ Gx)':

Derive: (∀x) (Fx ≡ Gx), ~ (∀x) (Fx ≡ Gx)

1	(∀x) (Fx ≡ Gx)	Assumption
2	(∃y) (Fy & ~ Gy)	Assumption
G	~ (∀x) (Fx ≡ Gx)	
G	(∀x) (Fx ≡ Gx)	1 R

The second assumption suggests using Existential Elimination, and we know it is wise to do as much of the work of the derivation as possible within the Existential Elimination subderivation:

Derive: (∀x)(Fx ≡ Gx), ~ (∀x)(Fx ≡ Gx)

1	(∀x)(Fx ≡ Gx)	Assumption
2	(∃y)(Fy & ~ Gy)	Assumption
3	Fa & ~ Ga	A / ∃E
G	~ (∀x)(Fx ≡ Gx)	
G	~ (∀x)(Fx ≡ Gx)	2, 3–___ ∃E
	(∀x)(Fx ≡ Gx)	1 R

Our current goal is a negation, which we will try to derive using Negation Introduction. We assume '(∀x)(Fx ≡ Gx)' *even though that sentence is one of our primary assumptions and hence already accessible.* We assume it because Negation Introduction requires that we assume the sentence whose negation we wish to derive:

Derive: (∀x)(Fx ≡ Gx), ~ (∀x)(Fx ≡ Gx)

1	(∀x)(Fx ≡ Gx)	Assumption
2	(∃y)(Fy & ~ Gy)	Assumption
3	Fa & ~ Ga	A / ∃E
4	(∀x)(Fx ≡ Gx)	A / ~ I
G	~ (∀x)(Fx ≡ Gx)	4–___ ~ I
G	~ (∀x)(Fx ≡ Gx)	2, 3–___ ∃E
G	(∀x)(Fx ≡ Gx)	1 R

We are now finally in a position where we can work profitably from the "top down". From line 4 we can derive 'Fa ≡ Ga' by Biconditional Elimination; from line 3 we can derive 'Fa'; and then it is easy to derive both 'Ga' and '~ Ga':

Derive: (∀x)(Fx ≡ Gx), ~ (∀x)(Fx ≡ Gx)

1	(∀x)(Fx ≡ Gx)	Assumption
2	(∃y)(Fy & ~ Gy)	Assumption
3	Fa & ~ Ga	A / ∃E
4	(∀x)(Fx ≡ Gx)	A / ~ I
5	Fa ≡ Ga	4 ∀E
6	Fa	3 &E
7	Ga	5, 6 ≡E
8	~ Ga	3 &E
9	~ (∀x)(Fx ≡ Gx)	4–8 ~ I
10	~ (∀x)(Fx ≡ Gx)	2, 3–9 ∃E
11	(∀x)(Fx ≡ Gx)	1 R

Had we taken '(∃y)(Fy & ~ Gy)' and '~ (∃y)(Fy & ~ Gy)' as our **Q** and ~ **Q** we would have produced the following very similar derivation:

Derive: (∃y)(Fy & ~ Gy), ~ (∃y)(Fy & ~ Gy)

```
 1 | (∀x)(Fx ≡ Gx)                    Assumption
 2 | (∃y)(Fy & ~ Gy)                  Assumption
   |----------------
 3 | | Fa & ~ Ga                      A / ∃E
   | |----------
 4 | | | (∃y)(Fy & ~ Gy)              A / ~ I
   | | |----------------
 5 | | | Fa ≡ Ga                      1 ∀E
 6 | | | Fa                           3 &E
 7 | | | Ga                           5, 6 ≡E
 8 | | | ~ Ga                         3 &E
 9 | | ~ (∃y)(Fy & ~ Gy)              4–8 ~ I
10 | ~ (∃y)(Fy & ~ Gy)                2, 3–9 ∃E
11 | (∃y)(Fy & ~ Gy)                  2 R
```

We will next demonstrate that {(∀z)(Hz ⊃ (∃y)Gzy), (∃w)Hw, (∀x) ~ (∃y)Gxy} is inconsistent in *PD*. Though the set includes no negations, we can immediately derive one, say '~ (∃y)Gay', by applying Universal Elimination to '(∀x) ~ (∃y)Gxy'. So we will take '(∃y)Gay' and '~ (∃y)Gay' as our goals:

Derive: (∃y)Gay, ~ (∃y)Gay

```
 1 | (∀z)(Hz ⊃ (∃y)Gzy)               Assumption
 2 | (∃w)Hw                           Assumption
 3 | (∀x) ~ (∃y)Gxy                   Assumption
   |----------------
   |
 G | (∃y)Gay
   | ~ (∃y)Gay                        3 ∀E
```

Our assumptions include the existentially quantified sentence '(∃w)Hw', so we will try to derive '(∃y)Gay' by Existential Elimination—which means we will have to be careful to pick a constant other than 'a' as the instantiating constant in our Existential Elimination assumption:

Derive: (∃y)Gay, ~ (∃y)Gay

```
 1 | (∀z)(Hz ⊃ (∃y)Gzy)               Assumption
 2 | (∃w)Hw                           Assumption
 3 | (∀x) ~ (∃y)Gxy                   Assumption
   |----------------
 4 | | Hb                             A / ∃E
   | |---
   | |
 G | | (∃y)Gay
 G | (∃y)Gay                          2, 4–__ ∃E
   | ~ (∃y)Gay                        3 ∀E
```

There is a problem in the offing here. We used ‘b’ as the instantiating constant at line 4 because ‘a’ occurs in the sentence we hope to obtain by Existential Elimination, ‘(∃y)Gay’. This means that we will be able to obtain ‘(∃y)Gby’, but not ‘(∃y)Gay’ by applying Universal Elimination to line 1 (obtaining ‘Hb ⊃ (∃y)Gby’ and then doing Conditional Elimination). So we need an alternative strategy for obtaining our current goal, ‘(∃y)Gay’. We will use Negation Elimination:

```
Derive: (∃y)Gay, ~ (∃y)Gay

 1  | (∀z)(Hz ⊃ (∃y)Gzy)              Assumption
 2  | (∃w)Hw                          Assumption
 3  | (∀x) ~ (∃y)Gxy                  Assumption
    |---------------
 4  |  | Hb                           A / ∃E
    |  |---
 5  |  |  | ~ (∃y)Gay                 A / ~ E
    |  |  |----------
    |  |  |
    |  |  |
 G  |  | (∃y)Gay                      5–___ ~ E
 G  | (∃y)Gay                         2, 4–6 ∃E
    | ~ (∃y)Gay                       3 ∀E
```

We can now complete the derivation by deriving both ‘(∃y)Gby’ and ‘~ (∃y)Gby’ within the scope of the assumption on line 5, the first by the steps mentioned previously, the second by applying Universal Elimination to the sentence on line 3.

```
Derive: (∃y)Gay, ~ (∃y)Gay

 1  | (∀z) (Hz ⊃ (∃y)Gzy)             Assumption
 2  | (∃w)Hw                          Assumption
 3  | (∀x) ~ (∃y)Gxy                  Assumption
    |---------------
 4  |  | Hb                           A / ∃E
    |  |---
 5  |  |  | ~ (∃y)Gay                 A / ~ E
    |  |  |----------
 6  |  |  | Hb ⊃ (∃y)Gby              1 ∀E
 7  |  |  | (∃y)Gby                   4, 6 ⊃E
 8  |  |  | ~ (∃y)Gby                 3 ∀E
 9  |  | (∃y)Gay                      5–8 ~ E
10  | (∃y)Gay                         2, 4–9 ∃E
11  | ~ (∃y)Gay                       3 ∀E
```

The technique of using a negation strategy within an Existential Elimination subderivation, as we have just done, is useful as a way of generating a sentence that does not violate any of the restrictions on Existential Elimination. It is useful whenever we can see that some sentence and its negation are derivable within the Existential Elimination subderivation, but those sentences contain a constant that keeps us from moving either out from the Existential Elimination subderivation by Existential Elimination. In such a case we can always derive a sentence that does not contain the Existential Elimination subderivation's instantiating constant. We can do this by assuming the negation of the desired sentence and deriving the contradictory sentences within the negation elimination subderivation.

10.2E EXERCISES

Note: Here, as always, the *Student Solutions Manual* contains answers to all unstarred exercises. In addition, when an exercise is preceded by a number sign (#) the *Solutions Manual* contains a detailed account of how the derivation given in the *Solutions Manual* is constructed.

1. Construct derivations that establish the validity of the following arguments:

a. $(\forall y)[Fy \supset (Gy \,\&\, Hy)]$
———
$(\forall x)(Fx \supset Hx)$

*b. $(\forall x)(Fx \equiv Gx)$
$(\exists x)Fx$
———
$(\exists x)(Fx \,\&\, Gx)$

#c. $(\forall y)[Gy \supset (Hy \,\&\, Fy)]$
$(\exists x)Gx$
———
$(\exists z)Fz$

*d. $(\forall x)[Fx \supset (Gx \,\&\, Hx)]$
$(\exists y)(Fy \,\&\, Dy)$
———
$(\exists z)Gz$

e. $(\exists x)Fx \supset (\forall x)Gx$
Fa
$(\forall x)(Gx \supset Hx)$
———
$(\forall x)Hx$

*f. $(\forall y)[(Hy \,\&\, Fy) \supset Gy]$
$(\forall z)Fz$
———
$(\forall x)(Hx \supset Gx)$

g. $(\forall x)Fx \vee (\forall x)Gx$
———
$(\forall x)(Fx \vee Gx)$

*h. $(\forall x)(Dx \equiv \sim Gx)$
$(\forall y)(Gy \supset Hy)$
$(\exists z) \sim Hz$
———
$(\exists z)Dz$

#i. $(\forall x)(Fx \supset Hx)$
$(\forall y)(Gy \supset Hy)$
———
$(\forall y)[(Fy \vee Gy) \supset Hy]$

*j. $(\exists y)(Fy \vee Gy)$
$(\forall x)(Fx \supset Hx)$
$(\forall x)(Gx \supset Hx)$
———
$(\exists z)Hz$

k. $(\exists x)Hx$
$(\forall x)(Hx \supset Rx)$
$(\exists x)Rx \supset (\forall x)Gx$
———
$(\forall x)(Fx \supset Gx)$

*l. $\sim (\exists x)Fx \equiv (\forall y)Gy$
$(\forall y) \sim Fy$
———
$(\exists y)Gy$

m. $(\forall x)Fx \vee (\forall y) \sim Gy$
$Fa \supset Hb$
$\sim Gb \supset Jb$
———
$(\exists y)(Hy \vee Jy)$

*n. $Fa \vee (\forall x) \sim Fx$
$(\exists y)Fy$
———
Fa

2. Prove that the following sentences of *PL* are theorems of *PD:*

a. Fa ⊃ (∃y)Fy
*b. (∀x)Fx ⊃ (∃y)Fy
c. (∀x)[Fx ⊃ (Gx ⊃ Fx)]
*d. ~ Fa ⊃ ~ (∀x)Fx
e. ~ (∃x)Fx ⊃ (∀x) ~ Fx
*f. (∃x)(∃y)Fxy ⊃ (∃y)(∃x)Fyx
g. Fa ∨ (∃y) ~ Fy
*h. (∀x)(Hx ⊃ Ix) ⊃ [(∃x)Hx ⊃ (∃x)Ix]
#i. [(∀x)Fx ∨ (∀x)Gx] ⊃ (∀x)(Fx ∨ Gx)
*j. [(∀x)Fx & (∃y)Gy] ⊃ (∃x)(Fx & Gx)
k. (∃x)(Fx & Gx) ⊃ [(∃x)Fx & (∃x)Gx]
*l. [(∃x)Fx ∨ (∃x)Gx] ⊃ (∃x)(Fx ∨ Gx)
m. (∀x)Hx ≡ ~ (∃x) ~ Hx

3. Construct derivations that establish that the following pairs of sentences are equivalent in *PD:*

a.	(∀x)(Fx & Gx)	(∀x)Fx & (∀x)Gx
*b.	(∀x)(Fx ⊃ Ga)	(∃x)Fx ⊃ Ga
c.	(∀x)Fx	~ (∃x) ~ Fx
*d.	(∃y)(Fy & (∀x)Gx)	(∃y)(∀x)(Fy & Gx)
#e.	(∃x)Fx	~ (∀x) ~ Fx
*f.	(∃x)(Fx & ~ Gx)	~ (∀x)(Fx ⊃ Gx)
g.	(∀z)(Hz ⊃ ~ Iz)	~ (∃y)(Hy & Iy)
*h.	(∃x)(Fa ⊃ Gx)	Fa ⊃ (∃x)Gx
i.	(∀x)(∃y)(Fx ⊃ Gy)	(∀x)(Fx ⊃ (∃y)Gy)

4. Construct derivations that establish that the following sets are inconsistent in *PD:*

a. {(∀x)(Fx ≡ ~ Fx)}
*b. {(∀x)Hx, (∀y) ~ (Hy ∨ Gyy)}
#c. {~ (∀x)Fx, ~ (∃x) ~ Fx}
*d. {~ (∀x) ~ Fx, ~ (∃x)Fx}
e. {(∀x)(Fx ⊃ Gx), (∃x)Fx, ~ (∃x)Gx}
*f. {(∀z) ~ Fz, (∃z)Fz}
g. {(∀x)Fx, (∃y) ~ Fy}
*h. {(∃y)(Hy & Jy), (∀x) ~ Jx}
i. {(∀x)(Hx ≡ ~ Gx), (∃x)Hx, (∀x)Gx}
*j. {(∀z)(Hz ⊃ Iz), (∃y)(Hy & ~ Iy)}
k. {(∀z)[Rz ⊃ (Tz & ~ Mz)], (∃y)(Ry & My)}
*l. {(∀x)(Fx ⊃ Gx), (∀x)(Fx ⊃ ~ Gx), (∃x)Fx}

5. Construct derivations that establish the following:

a. {(∃y)(∀x)Fxy} ⊢ (∀x)(∃y)Fxy
*b. {(∀z)(Gz ⊃ (∃x)Fxz), (∀x)Gx} ⊢ (∀z)(∃x)Fxz
c. {(∃x)Fxxx} ⊢ (∃x)(∃y)(∃z)Fxyz
*d. {(∀x)(∀y)(Bx ⊃ Txy} ⊢ (∀x)(∀y)[(Bx & Ny) ⊃ Txy]
e. {(∀x)(Fx ⊃ (∃y)Gxy), (∃x)Fx} ⊢ (∃x)(∃y)Gyx
*f. {(∀x)(∃y)Gxy, (∀x)(∀y)(Hxy ⊃ ~ Gxy)} ⊢ (∀x)(∃z) ~ Hxz
g. {(∀x)(∀y)(Hxy ⊃ ~ Hyx), (∃x)(∃y)Hxy} ⊢ (∃x)(∃y) ~ Hyx
*h. {(∀x)(∀y)Fxy ∨ (∀x)(∀y)Gxy} ⊢ (∀x)(∀y)(Fxy ∨ Gxy)
i. {~ (∃x)(∃y)Rxy, (∀x)(∀y)(~ Hxy ≡ Rxy)} ⊢ (∀x)(∀y)Hxy
*j. {(∀x)(∀y)(Fxy ≡ ~ Gyx), (∃z)(∃w)Gzw} ⊢ (∃x)(∃y) ~ Fxy

6. Construct derivations that establish the validity of the following arguments:

a.
$$\begin{array}{l} (\forall x)(Fx \supset Gba) \\ (\exists x)Fx \\ \hline (\exists y)Gya \end{array}$$

*b.
$$\begin{array}{l} (\forall x)(Hx \supset (\forall y)Rxyb) \\ (\forall x)(\forall z)(Razx \supset Sxzz) \\ \hline Ha \supset (\exists x)Sxcc \end{array}$$

c.
$$\begin{array}{l} (\exists x)(\exists y)(Fxy \vee Fyx) \\ \hline (\exists x)(\exists y)Fxy \end{array}$$

*d.
$$\begin{array}{l} (\forall x)(Fxa \supset Fax) \\ (\exists x)(Hx \;\&\; \sim Fax) \\ \hline \sim (\forall y)(Hy \supset Fya) \end{array}$$

e.
$$\begin{array}{l} (\forall x)(\forall y)[(\exists z)(Fyz \;\&\; \sim Fzx) \supset Gxy] \\ \sim (\exists x)Gxx \\ \hline (\forall z)(Faz \supset Fza) \end{array}$$

*f.
$$\begin{array}{l} (\forall x)(\forall y)(Dxy \supset Cxy) \\ (\forall x)(\exists y)Dxy \\ (\forall x)(\forall y)(Cxy \supset Cyx) \\ \hline (\exists x)(\exists y)(Cxy \;\&\; Cyx) \end{array}$$

g.
$$\begin{array}{l} (\forall x)(Fx \supset (\exists y)Gxy \\ (\forall x)(\forall y) \sim Gxy \\ \hline (\forall x) \sim Fx \end{array}$$

*h.
$$\begin{array}{l} (\forall x)(Fx \supset (\exists y)Gxy) \\ (\forall x)(\forall y)(Gxy \supset Hxy) \\ \sim (\exists x)(\exists y)Hxy \\ \hline \sim (\exists x)Fx \end{array}$$

7. Prove that the following sentences of *PL* are theorems of *PD:*

a. $(\forall x)(\exists z)(Fxz \supset Fzx)$
*b. $(\forall x)Fxx \supset (\forall x)(\exists y)Fxy$
c. $(\forall x)(\forall y)Gxy \supset (\forall z)Gzz$
*d. $(\exists x)Fxx \supset (\exists x)(\exists y)Fxy$
e. $(\forall x)Lxx \supset (\exists x)(\exists y)(Lxy \;\&\; Lyx)$
*f. $(\exists x)(\forall y)Lxy \supset (\exists x)Lxx$
#g. $(\exists x)(\forall y)Fxy \supset (\exists x)(\exists y)Fxy$
*h. $(\forall x)(Fx \supset (\exists y)Gya) \supset (Fb \supset (\exists y)Gya)$
i. $(\exists x)(\exists y)(Lxy \equiv Lyx)$
*j. $(\exists x)(\forall y)Hxy \supset (\forall y)(\exists x)Hxy$
k. $(\forall x)(\forall y)(\forall z)Gxyz \supset (\forall x)(\forall y)(\forall z)(Gxyz \supset Gzyx)$
*l. $(\forall x)(Fx \supset (\exists y)Gyx) \supset ((\exists x)Fx \supset (\exists x)(\exists y)Gxy)$
m. $(\forall x)(\forall y)(Fxy \equiv Fyx) \supset \sim (\exists x)(\exists y)(Fxy \;\&\; \sim Fyx)$
*n. $(\exists x)(Fx \supset (\forall y)Fy)$

8. Construct derivations that establish that the following pairs of sentences are equivalent in *PD:*

a.	$(\forall x)(Fx \supset (\exists y)Gya)$	$(\exists x)Fx \supset (\exists y)Gya$
*b.	$(\forall x)(Fx \supset (\forall y)Gy)$	$(\forall x)(\forall y)(Fx \supset Gy)$
#c.	$(\exists x)[Fx \supset (\forall y)Hxy]$	$(\exists x)(\forall y)(Fx \supset Hxy)$
*d.	$(\forall x)(\forall y)(Fxy \supset Gy)$	$(\forall y)[(\exists x)Fxy \supset Gy]$
e.	$(\forall x)(\forall y)(Fxy \equiv \sim Gyx)$	$(\forall x)(\forall y) \sim (Fxy \equiv Gyx)$

9. Construct derivations that establish that the following sets are inconsistent in *PD:*

a. $\{(\forall x)(\forall y)[(Ex \;\&\; Ey) \supset Txy],\ (Ea \;\&\; Eb) \;\&\; \sim Tab\}$
*b. $\{(\forall x)(\exists y)Lyx,\ \sim (\exists x)Lxb\}$

c. {~ (∃x)Fxx, (∃x)(∀y)Fxy}
*d. {(∀x)(∀y)(Fxy ⊃ Fyx), Fab, ~ (∃z)Fza}
e. {(∀x)(∃y)Lxy, (∀y) ~ Lay}
*f. {(∃x)(∀y)Gxy, ~ (∀y)(∃x)Gxy}
g. {(∀x)[Hx ⊃ (∃y)Lyx], (∃x) ~ (∃y)Lyx, (∀x)Hx}
*h. {~ (∃x)Fxx, (∀x)[(∃y)Fxy ⊃ Fxx], (∃x)(∃y)Fxy}
#i. {(∀x)(∃y)Fxy, (∃z) ~ (∃w)Fzw}
*j. {(∀x)(∀y)(Gxy ≡ Gyx), (∃x)(∃y)(Gxy & ~ Gyx)}
k. {(∀x)(∀y)(Fxy ∨ Gxy), (∃x)(∃y)(~ Fxy & ~ Gxy)}
*l. {(∀x)(Fx ⊃ [(∃y)Gy ⊃ (∀y)Gy]), (∃x)(Fx & Gx), (∃y) ~ Gy}

10.3 THE DERIVATION SYSTEM *PD+*

PD+ is a derivation system that includes all the rules of *PD,* the rules that distinguish *SD+* from *SD,* and one additional rule of replacement. *PD+* is no stronger than *PD;* however, derivations in *PD+* are often shorter than the corresponding derivations in *PD*. The rules of replacement in *PD+* apply to subformulas of sentences as well as to complete sentences. In the following example each of the replacement rules has been applied to a subformula of the sentence on the previous line:

1	(∀x)[(Fx & Hx) ⊃ (∃y)Nxy]	Assumption
2	(∀x)[~ (Fx & Hx) ∨ (∃y)Nxy]	1 Impl
3	(∀x)[~ (Fx & Hx) ∨ ~ ~ (∃y)Nxy]	2 DN
4	(∀x) ~ [(Fx & Hx) & ~ (∃y)Nxy]	3 DeM
5	(∀x) ~ [(Hx & Fx) & ~ (∃y)Nxy]	4 Com

Here Implication was applied to the subformula '(Fx & Hx) ⊃ (∃y)Nxy' of the sentence on line 1 to produce the subformula '~ (Fx & Hx) ∨ (∃y)Nxy' of the sentence on line 2. Double Negation was applied to the subformula '(∃y)Nxy' of the sentence on line 2, to produce the subformula '~ ~ (∃y)Nxy' of the sentence on line 3. De Morgan was applied to the subformula '~ (Fx & Hx) ∨ ~ ~ (∃y) Nxy' of the sentence on line 3 to produce the subformula '~ [(Fx & Hx) & ~ (∃y)Nxy]' of the sentence on line 4. Finally, Commutation was applied to the subformula 'Fx & Hx' of the sentence on line 4 to produce the 'Hx & Fx' of the sentence on line 5.

In applying rules of replacement in *PD+* it is important to correctly identify subformulas of sentences. Consider the following:

1	(∀x)[Lx ∨ (∃y)(Bxy ∨ Jxy)]	Assumption	
2	(∀x)[(Lx ∨ (∃y)Bxy) ∨ Jxy]	1 Assoc	**MISTAKE!**

Line 2 is a mistake because the immediate subformula of the sentence on line 1 is not of the form **P** ∨ (**Q** ∨ **R**). Rather, it is of the form **P** ∨ (∃**x**)(**Q** ∨ **R**).

In addition to the rules of replacement of *SD+*, *PD+* contains **Quantifier Negation.** Where **P** is an open sentence of *PL* in which **x** occurs free, the rule is

Quantifier Negation (QN)

~(∀**x**)**P** ◁▷ (∃**x**) ~ **P**
~(∃**x**)**P** ◁▷ (∀**x**) ~ **P**

As with all rules of replacement, Quantifier Negation can be applied to subformulas within a sentence, as well as to an entire sentence. All these are proper uses of Quantifier Negation:

1	~ (∃y) ~ (∀x) (Fx ⊃ (∃z) ~ Gxy)	Assumption
2	(∀y) ~ ~ (∀x) (Fx ⊃ (∃z) ~ Gxy)	1 QN
3	(∀y) ~ (∃x) ~ (Fx ⊃ (∃z) ~ Gxy)	2 QN
4	(∀y) ~ (∃x) ~ (Fx ⊃ ~ (∀z) Gxy)	3 QN

The definitions of the basic concepts of *PD+* strictly parallel the definitions of the basic concepts of *PD,* in all cases replacing '*PD*' with '*PD+*'. Consequently the tests for the various syntactic properties are carried out in the same way. The important difference between *PD* and *PD+* is that *PD,* with fewer rules, provides theoretical elegance and *PD+*, with more rules, provides practical ease.

In Section 10.2 we proved that '(∃x) (Fx ⊃ (∀y) Fy)' is a theorem in *PD*. Our derivation was 17 lines long. We repeat it here.

Derive: (∃x) (Fx ⊃ (∀y) Fy)

1	~ (∃x) (Fx ⊃ (∀y) Fy)	A / ~ E
2	Fa	A / ⊃I
3	~ Fb	A / ~ I
4	Fb	A / ⊃I
5	~ (∀y) Fy	A / ~ E
6	Fb	4 R
7	~ Fb	3 R
8	(∀y) Fy	5–7 ~ E
9	Fb ⊃ (∀y) Fy	4–8 ⊃I
10	(∃x) (Fx ⊃ (∀y) Fy)	9 ∃I
11	~ (∃x) (Fx ⊃ (∀y) Fy)	1 R
12	Fb	3–11 ~ E
13	(∀y) Fy	12 ∀I
14	Fa ⊃ (∀y) Fy	2–13 ⊃I
15	(∃x) (Fx ⊃ (∀y) Fy)	14 ∃I
16	~ (∃x) (Fx ⊃ (∀y) Fy)	1 R
17	(∃x) (Fx ⊃ (∀y) Fy)	1–16 ~ E

We can show that this sentence is a theorem in *PD+* in just 10 lines:

Derive: (∃x) (Fx ⊃ (∀y)Fy)

1	~ (∃x) (Fx ⊃ (∀y)Fy)	A / ~ E
2	(∀x) ~ (Fx ⊃ (∀y)Fy)	1 QN
3	~ (Fa ⊃ (∀y)Fy)	2 ∀E
4	~ (~ Fa ∨ (∀y)Fy)	3 Impl
5	~ ~ Fa & ~ (∀y)Fy	4 DeM
6	~ ~ Fa	5 &E
7	Fa	6 DN
8	~ (∀y)Fy	5 &E
9	(∀y)Fy	7 ∀I
10	(∃x) (Fx ⊃ (∀y)Fy)	1–9 ~ E

In Section 10.2 it took us 19 lines to derive '(∃x) (Fx ⊃ Ga)' from {(∀x)Fx ⊃ Ga}. We repeat our derivation here:

Derive: (∃x) (Fx ⊃ Ga)

1	(∀x)Fx ⊃ Ga	Assumption
2	~ (∃x) (Fx ⊃ Ga)	A / ~ E
3	Fa	A / ⊃I
4	~ Fb	A / ~ E
5	Fb	A / ⊃I
6	~ Ga	A / ~ E
7	Fb	5 R
8	~ Fb	4 R
9	Ga	6–8 ~ E
10	Fb ⊃ Ga	5–9 ⊃I
11	(∃x) (Fx ⊃ Ga)	10 ∃I
12	~ (∃x) (Fx ⊃ Ga)	2 R
13	Fb	4–12 ~ E
14	(∀x)Fx	13 ∀I
15	Ga	1, 14 ⊃E
16	Fa ⊃ Ga	3–15 ⊃I
17	(∃x) (Fx ⊃ Ga)	16 ∃I
18	~ (∃x) (Fx ⊃ Ga)	1 R
19	(∃x) (Fx ⊃ Ga)	1–18 ~ E

We can derive '(∃x) (Fx ⊃ Ga)' from {(∀x)Fx ⊃ Ga} in just 12 lines in *PD+*:

Derive: (∃x) (Fx ⊃ Ga)

1	(∀x)Fx ⊃ Ga	Assumption
2	~ (∃x) (Fx ⊃ Ga)	A / ~ E
3	(∀x) ~ (Fx ⊃ Ga)	2 QN
4	~ (Fb ⊃ Ga)	3 ∀E
5	~ (~ Fb ∨ Ga)	4 Impl
6	~ ~ Fb & ~ Ga	5 DeM
7	~ ~ Fb	6 &E
8	Fb	7 DN
9	(∀x)Fx	8 ∀I
10	Ga	1, 9 ⊃E
11	~ Ga	6 &E
12	(∃x) (Fx ⊃ Ga)	2–11 ~ E

10.3E EXERCISES

1. Show that each of the following derivability claims holds in *PD+*.

a. {~ (∀y) (Fy & Gy)} ⊢ (∃y) (~ Fy ∨ ~ Gy)
*b. {(∀w) (Lw ⊃ Mw), (∀y) (My ⊃ Ny)} ⊢ (∀w) (Lw ⊃ Nw)
c. {(∃z) (Gz & Az), (∀y) (Cy ⊃ ~ Gy)} ⊢ (∃z) (Az & ~ Cz)
*d. {~ (∃x) (~ Rx & Sxx), Sjj} ⊢ Rj
e. {(∀x) [(~ Cxb ∨ Hx) ⊃ Lxx], (∃y) ~ Lyy} ⊢ (∃x)Cxb
*f. {(∀x)Fx, (∀z)Hz} ⊢ ~ (∃y) (~ Fy ∨ ~ Hy)

2. Show that each of the following arguments is valid in *PD+*.

a. (∀x) ~ Jx
(∃y) (Hby ∨ Ryy) ⊃ (∃x)Jx
———
(∀y) ~ (Hby ∨ Ryy)

*b. ~ (∃x) (∀y) (Pxy & ~ Qxy)
———
(∀x) (∃y) (Pxy ⊃ Qxy)

c. (∀x) ~ ((∀y)Hyx ∨ Tx)
~ (∃y) (Ty ∨ (∃x) ~ Hxy)
———
(∀x) (∀y)Hxy & (∀x) ~ Tx

*d. (∀z) (Lz ≡ Hz)
(∀x) ~ (Hx ∨ ~ Bx)
———
~ Lb

e. (∀z) [Kzz ⊃ (Mz & Nz)]
(∃z) ~ Nz
———
(∃x) ~ Kxx

*f. $(\exists x)[\sim Bxm \ \& \ (\forall y)(Cy \supset \sim Gxy)]$
$(\forall z)[\sim (\forall y)(Wy \supset Gzy) \supset Bzm]$
―――――――――――――
$(\forall x)(Cx \supset \sim Wx)$

g. $(\exists z)Qz \supset (\forall w)(Lww \supset \sim Hw)$
$(\exists x)Bx \supset (\forall y)(Ay \supset Hy)$
―――――――――――――
$(\exists w)(Qw \ \& \ Bw) \supset (\forall y)(Lyy \supset \sim Ay)$

*h. $(\forall y)(Kby \supset \sim Hy)$
―――――――――――――
$(\forall x)[(\exists y)(Kby \ \& \ Qxy) \supset (\exists z)(\sim Hz \ \& \ Qxz)]$

i. $\sim (\forall x)(\sim Gx \vee \sim Hx) \supset (\forall x)[Cx \ \& \ (\forall y)(Ly \supset Axy)]$
$(\exists x)[Hx \ \& \ (\forall y)(Ly \supset Axy)] \supset (\forall x)(Fx \ \& \ (\forall y)Bxy)$
―――――――――――――
$\sim (\forall x)(\forall y)Bxy \supset (\forall x)(\sim Gx \vee \sim Hx)$

3. Show that each of the following sentences is a theorem in *PD+*.
a. $(\forall x)(Ax \supset Bx) \supset (\forall x)(Bx \vee \sim Ax)$
*b. $(\forall x)(Ax \supset (Ax \supset Bx)) \supset (\forall x)(Ax \supset Bx)$
c. $\sim (\exists x)(Ax \vee Bx) \supset (\forall x) \sim Ax$
*d. $(\forall x)(Ax \supset Bx) \vee (\exists x)Ax$
e. $((\exists x)Ax \supset (\exists x)Bx) \supset (\exists x)(Ax \supset Bx)$
*f. $(\forall x)(\exists y)(Ax \vee By) \equiv (\exists y)(\forall x)(Ax \vee By)$

4. Show that the members of each of the following pairs of sentences are equivalent in *PD+*.

a.	$\sim (\forall x)(Ax \supset Bx)$	$(\exists x)(Ax \ \& \sim Bx)$
*b.	$(\exists x)(\exists y)Axy \supset Aab$	$(\exists x)(\exists y)Axy \equiv Aab$
c.	$\sim (\forall x) \sim [(Ax \ \& \ Bx) \supset Cx]$	$(\exists x)[\sim Ax \vee (\sim Cx \supset \sim Bx)]$
*d.	$\sim (\forall x)(\exists y)[(Ax \ \& \ Bx) \vee Cy]$	$(\exists x)(\forall y)[\sim (Cy \vee Ax) \vee \sim (Cy \vee Bx)]$
e.	$(\forall x)(Ax \equiv Bx)$	$\sim (\exists x)[(\sim Ax \vee \sim Bx) \ \& \ (Ax \vee Bx)]$
*f.	$(\forall x)(Ax \ \& \ (\exists y) \sim Bxy)$	$\sim (\exists x)[\sim Ax \vee (\forall y)(Bxy \ \& \ Bxy)]$

5. Show that each of the following sets of sentences is inconsistent in *PD+*.
a. $\{[(\forall x)(Mx \equiv Jx) \ \& \sim Mc] \ \& \ (\forall x)Jx\}$
*b. $\{\sim Fa, \sim (\exists x)(\sim Fx \vee \sim Fx)\}$
c. $\{(\forall x)(\forall y)Lxy \supset \sim (\exists z)Tz, (\forall x)(\forall y)Lxy \supset ((\exists w)Cww \vee (\exists z)Tz),$
$(\sim (\forall x)(\forall y)Lxy \vee (\forall z)Bzzk) \ \& \ (\sim (\forall z)Bzzk \vee \sim (\exists w)Cww), (\forall x)(\forall y)Lxy\}$
*d. $\{(\exists x)(\forall y)(Hxy \supset (\forall w)Jww), (\exists x) \sim Jxx \ \& \sim (\exists x) \sim Hxm\}$
e. $\{(\forall x)(\forall y)(Gxy \supset Hc), (\exists x)Gix \ \& \ (\forall x)(\forall y)(\forall z)Lxyz, \sim Lcib \vee \sim (Hc \vee Hc)\}$
*f. $\{(\forall x)[(Sx \ \& \ Bxx) \supset Kax], (\forall x)(Hx \supset Bxx), (\exists x)(Sx \ \& \ Hx),$
$(\forall x) \sim (Kax \ \& \ Hx)\}$

6. a. Show that Universal Introduction and Universal Elimination are eliminable in *PD+* by developing routines that can be used in place of these rules to obtain the same results. (*Hint:* Consider using Quantifier Negation, Existential Introduction, and Existential Elimination.)
*b. Show that Existential Introduction and Existential Elimination are eliminable in *PD+* by developing routines that can be used in place of these rules to obtain the same results. (*Hint:* Consider using Quantifier Negation, Universal Introduction, and Universal Elimination.)

The symbolic language *PLE* extends *PL* to include sentences that contain functors and the identity predicate. Accordingly we need to extend the derivation system *PD* developed earlier in this chapter to allow for derivations that include these new sentences of *PLE*. We shall do so by adding an introduction rule and an elimination rule for the identity predicate, and then modifying the quantifier rules so as to allow for sentences containing functors. The resulting *extended* predicate derivation system is called *PDE*.

The introduction rule for '=' is

Identity Introduction (=I)

$\triangleright \mid (\forall \mathbf{x})\mathbf{x} = \mathbf{x}$

Identity Introduction is unlike other introduction rules in that it appeals to no previous line or lines of the derivation. Rather, it allows sentences of the specified form to be entered on any line of any derivation, no matter what sentences, if any, occur earlier in the derivation.[1] Identity Introduction is truth-preserving because every sentence that can be introduced by it, that is every sentence of the form $(\forall \mathbf{x})\mathbf{x} = \mathbf{x}$, is quantificationally true. These sentences simply say of each thing that it is identical to itself. Here is a very simple derivation of a theorem using the rule Identity Introduction:

Derive: a = a

1	$(\forall y)y = y$	=I
2	$a = a$	1 ∀E

Notice that the sentence on line 1 is *not an assumption.*

The elimination rule for "=" is

Identity Elimination (=E)

$$\begin{array}{l|l} & \mathbf{t}_1 = \mathbf{t}_2 \\ & \mathbf{P} \\ \triangleright & \mathbf{P}(\mathbf{t}_1//\mathbf{t}_2) \end{array} \quad \text{or} \quad \begin{array}{l|l} & \mathbf{t}_1 = \mathbf{t}_2 \\ & \mathbf{P} \\ \triangleright & \mathbf{P}(\mathbf{t}_2//\mathbf{t}_1) \end{array}$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are closed terms.

The notation

$$\mathbf{P}(\mathbf{t}_1//\mathbf{t}_2)$$

is read '**P** with one or more occurrences of $\mathbf{t}_2$ replaced by $\mathbf{t}_1$'. Similarly $\mathbf{P}(\mathbf{t}_2//\mathbf{t}_1)$ is read '**P** with one or more occurrences of $\mathbf{t}_1$ replaced by $\mathbf{t}_2$'. Recall that the closed terms of *PLE* are the individual constants together with complex terms

[1] Metaformulas (such as '$(\forall \mathbf{x})\mathbf{x}=\mathbf{x}$') that specify sentences that can be introduced without reference to previous sentences occurring in a derivation are usually called **axiom schemas.** An axiom schema is a metaformula such that every formula having its form may be entered in a derivation. Some derivation systems rely primarily on axiom schemas; these are called **axiomatic systems.**

such as '*f*(a,b)' and '*f*(*g*(a,b),c)' that contain no variables. Identity Elimination permits the replacement of one closed term with another in a sentence only if those closed terms designate the same thing ($\mathbf{t}_1 = \mathbf{t}_2$ says that $\mathbf{t}_1$ and $\mathbf{t}_2$ do designate the same thing). The following simple examples illustrate the use of this rule:

Derive: Hda

1	c = d	Assumption
2	Hca	Assumption
3	Hda	1, 2 =E

The following three derivations are very similar but not identical:

Derive: $(\forall x)(Fxh \supset Ghx)$

1	h = e	Assumption
2	$(\forall y)(Fye \supset Gey)$	Assumption
3	$(\forall y)(Fyh \supset Ghy)$	1, 2 =E
4	$Fah \supset Gha$	3 $\forall$E
5	$(\forall x)(Fxh \supset Ghx)$	4 $\forall$I

Derive: $(\forall x)(Fxe \supset Ghx)$

1	h = e	Assumption
2	$(\forall y)(Fye \supset Gey)$	Assumption
3	$(\forall y)(Fye \supset Ghy)$	1, 2 =E
4	$Fae \supset Gha$	3 $\forall$E
5	$(\forall x)(Fxe \supset Ghx)$	4 $\forall$I

Derive: $(\forall x)(Fxh \supset Gex)$

1	h = e	Assumption
2	$(\forall y)(Fye \supset Gey)$	Assumption
3	$(\forall y)(Fyh \supset Gey)$	1, 2 =E
4	$Fah \supset Gea$	3 $\forall$E
5	$(\forall x)(Fxh \supset Gex)$	4 $\forall$I

In the first derivation we replaced, at line 3, both occurrences of 'e' in line 2 with 'h'. In the second derivation we replaced, at line 3, only the second occurrence of 'e' in line 2 with 'h'. And in the third derivation we replaced, at line 3, only the first occurrence of 'e' in line 2 with 'h'. All of these are appropriate uses of Identity Elimination, as are the following:

Derive: Hc

1	$(\forall x)Hf(a,x)$	Assumption
2	$c = f(a,b)$	Assumption
3	$Hf(a,b)$	1 $\forall$E
4	Hc	2, 3 =E

Derive: Wab

1	Haa ⊃ Waa	Assumption
2	Hab	Assumption
3	a = b	Assumption
4	Hab ⊃ Wab	1, 3 =E
5	Wab	2, 4 ⊃E

Note that from lines 1 through 3 we can obtain, by Identity Elimination, not just 'Hab ⊃ Wab' but a host of additional sentences, including those on lines 5 through 8 below:

1	Haa ⊃ Waa	Assumption
2	Hab	Assumption
3	a = b	Assumption
4	Hab ⊃ Wab	1, 3 =E
5	Hbb ⊃ Wbb	1, 3 =E
6	Haa ⊃ Wbb	1, 3 =E
7	Hbb ⊃ Waa	1, 3 =E
8	Hba ⊃ Wba	1, 3 =E

But these additional sentences do not advance us toward our goal of 'Wab'. There are alternative ways of deriving 'Wab'. Here is one:

Derive: Wab

1	Haa ⊃ Waa	Assumption
2	Hab	Assumption
3	a = b	Assumption
4	Haa	2, 3 =E
5	Waa	1, 4 ⊃E
6	Wab	3, 5 =E

Consider next these derivations:

Derive: Had

1	c = d	Assumption
2	Hac	Assumption
3	Had	1, 2 =E

Derive: (∀x)(Fxh ⊃ Ghx)

1	h = e	Assumption
2	(∀y)(Fye ⊃ Gey)	Assumption
3	(∀y)(Fyh ⊃ Ghy)	1, 2 =E
4	Fah ⊃ Gha	3 ∀E
5	(∀x)(Fxh ⊃ Ghx)	4 ∀I

Derive: Hc

1	$(\forall x)Hf(a,x)$	Assumption
2	$c = f(a,b)$	Assumption
3	$Hf(a,b)$	1 ∀E
4	Hc	2, 3 =E

The sentence '(a = b & b = c) ⊃ a = c' says that if a is identical to b, and b is identical to c, then a is identical to c. As expected, it is a theorem of *PDE*. Here is a proof:

Derive: (a = b & b = c) ⊃ a = c

1	a = b & b = c	A / ⊃I
2	a = b	1 &E
3	b = c	1 &E
4	a = c	2, 3 =E
5	(a = b & b = c) ⊃ a = c	1–4 ⊃I

In considering this example one might well ask whether the justification for line 4 indicates that we have replaced 'b' in line 3 with 'a', based on the identity at line 2, or that we replaced 'b' in line 2 with 'c' based on the identity at line 3. Fortunately, both replacements are allowed so the justification can be understood either way.

As we have already seen, sentences of the form $\mathbf{t}_1 = \mathbf{t}_1$ are normally obtained by Identity Introduction, as in

1	b = b ⊃ Fb	Assumption
2	(∀x)x = x	=I
3	b = b	2 ∀E
4	Fb	1, 3 ⊃E

In special circumstances we can obtain a sentence of the form **a** = **a** by Identity Elimination. This happens when the **a** of **a** = **a** already occurs in an accessible identity sentence. Here is an example:

1	b = b ⊃ Fb	Assumption
2	a = b	Assumption
3	b = b	2, 2 =E
4	Fb	1, 3 ⊃E

Identity Elimination allows us, given a sentence of the form $\mathbf{t}_1 = \mathbf{t}_2$, to replace any occurrence of $\mathbf{t}_1$ with $\mathbf{t}_2$ in any sentence that contains $\mathbf{t}_1$, and vice versa. In our example we have the identity sentence 'a = b' and that very sentence contains 'a', so we can replace the 'a' in 'a = b' with 'b', and we do so at line 3.

As we saw in Chapter 7, the identity predicate is useful in symbolizing sentences containing definite descriptions. Consider the argument:

The Roman general who defeated Pompey conquered Gaul.
Julius Caesar is a Roman general, and he defeated Pompey.

Julius Caesar conquered Gaul.

This argument can be symbolized in *PLE* as:

$(\exists x)[((Rx \ \& \ Dxp) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = x]) \ \& \ Cxg]$
$Rj \ \& \ Djp$

Cjg

This argument is valid, for if there is one and only one thing that is a Roman general and defeated Pompey, and if Julius Caesar is a Roman general who defeated Pompey, then Caesar is *the* Roman general who defeated Pompey, and is therefore someone who conquered Gaul. We can show this argument is valid in *PDE:*

Derive: Cjg

1	$(\exists x)[((Rx \ \& \ Dxp) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = x]) \ \& \ Cxg]$	Assumption
2	$Rj \ \& \ Djp$	Assumption
3	$((Ra \ \& \ Dap) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = a]) \ \& \ Cag$	A / ∃E
4	$(Ra \ \& \ Dap) \ \& \ (\forall y)[(Ry \ \& \ Dyp) \supset y = a]$	3 &E
5	$(\forall y)[(Ry \ \& \ Dyp) \supset y = a]$	4 &E
6	$(Rj \ \& \ Djp) \supset j = a$	5 ∀E
7	$j = a$	2, 6 ⊃E
8	Cag	3 &E
9	Cjg	7, 8 =E
10	Cjg	1, 3–9 ∃E

Here is another argument that involves a definite description.

The primary author of the Declaration of Independence was a slave owner.
Thomas Jefferson was the primary author of the Declaration of Independence.

Thomas Jefferson was a slave owner.

The conclusion of this argument can be symbolized as ‘Ot’ where ‘Ox’ is interpreted as ‘x owns at least one slave’ and ‘t’ designates Thomas Jefferson. To symbolize the premises we need a way of saying there was one and only one primary author of the Declaration of Independence. We can do so as follows:

$(\exists x)[Px \ \& \ (\forall z)(Pz \supset z = x)]$

We are here using 'Px' for 'x is a primary author of the Declaration of Independence'. This sentence of *PL* can be read as 'There is at least one thing x that is a primary author of the Declaration of Independence and each thing z that is a primary author of the Declaration of Independence is identical to x.' The full argument can now be symbolized as:

$$(\exists x)([Px \ \& \ (\forall z)(Pz \supset z = x)] \ \& \ Ox)$$
$$Pt \ \& \ (\forall z)(Pz \supset z = t)$$
$$\overline{}$$
$$Ot$$

We can construct a derivation that establishes that the above argument is valid in *PDE*. Here is a start:

Derive: Ot

1	$(\exists x)([Px \ \& \ (\forall z)(Pz \supset z = x)] \ \& \ Ox)$	Assumption
2	$Pt \ \& \ (\forall z)(Pz \supset z = t)$	Assumption
3	$\quad [Pa \ \& \ (\forall z)(Pz \supset z = a)] \ \& \ Oa$	A / ∃E
G	$\quad Ot$	
G	Ot	1, 2–__ ∃E

Our intent is to derive the final goal using Existential Elimination. If we can derive 'Ot' within the Existential Elimination subderivation we will be able to move it out of that subderivation because 't' is not the instantiating constant in our assumption at line 3 (it is for this reason that we picked a constant other than 't' as our instantiating constant at line 3). 'Oa' can be derived immediately from line 3 by Conjunction Elimination. What remains is to get to a point where we can use Identity Elimination to infer 'Ot' from 'Oa' and an appropriate identity sentence, either 'a = t' or 't = a'.

Derive: Ot

1	$(\exists x)([Px \ \& \ (\forall z)(Pz \supset z = x)] \ \& \ Ox)$	Assumption
2	$Pt \ \& \ (\forall z)(Pz \supset z = t)$	Assumption
3	$\quad [Pa \ \& \ (\forall z)(Pz \supset z = a)] \ \& \ Oa$	A / ∃E
4	$\quad Oa$	3 &E
G	$\quad a = t$	
G	$\quad Ot$	4, __ =E
G	Ot	1, 2–__ ∃E

Identity sentences are obtainable both from line 2 and from line 3. This suggests two strategies, and both will work. First we will try to obtain 'a = t'. We start by obtaining '(∀z)(Pz ⊃ z = t)' from line 2 by Conjuction Elimination and then 'Pa ⊃ a = t' by Universal Elimination. And 'Pa' is available from line 3 by two uses of Conjunction Elimination. This will allow us to complete the derivation:

Derive: Ot

1	(∃x)([Px & (∀z)(Pz ⊃ z = x) & Ox)	Assumption
2	Pt & (∀z)(Pz ⊃ z = t)	Assumption
3	[Pa & (∀z)(Pz ⊃ z = a)] & Oa	A / ∃E
4	Oa	3 &E
5	(∀z)(Pz ⊃ z = t)	2 &E
6	Pa ⊃ a = t	5 ∀E
7	Pa & (∀z)(Pz ⊃ z = a)	3 &E
8	Pa	7 &E
9	a = t	6, 8 ⊃E
10	Ot	4, 9 =E
11	Ot	1, 3–10 ∃E

We could also have completed our derivation by deriving the identity sentence 't = a' as follows:

Derive: Ot

1	(∃x)([Px & (∀z)(Pz ⊃ z = x)] & Ox	Assumption
2	Pt & (∀z)(Pz ⊃ z = t)	Assumption
3	[Pa & (∀z)(Pz ⊃ z = a)] & Oa	A / ∃E
4	Oa	3 &E
5	Pa & (∀z)(Pz ⊃ z = a)	3 &E
6	(∀z)(Pz ⊃ z = a)	5 &E
7	Pt ⊃ t = a	6 ∀E
8	Pt	2 &E
9	t = a	7, 8 ⊃E
10	Ot	4, 9 =E
11	Ot	1, 3–10 ∃E

When we formulated Identity Elimination we did so in a way that allows for the presence of complex terms in *PDE*. Two of our quantifier rules, Existential Introduction and Universal Elimination, need to be modified so that they too allow for the presence of complex terms. The other rules of *PD* function without modification as part of *PDE*. We recast Existential Introduction and Universal Elimination as follows:

Existential Introduction (∃I)

| **P**(**t**/**x**)

▷ | (∃**x**)**P**

where **t** is any closed term

Universal Elimination (∀E)

	(∀**x**)**P**
▷	**P**(**t**/**x**)

where **t** is any closed term

Consider the following simple derivations:

Derive: (∃z)Fz

1	(∀y)Fy	Assumption
2	Fa	1 ∀E
3	(∃z)Fz	2 ∃I

Derive: (∃z)F*g*(z)

1	(∀y)Fy	Assumption
2	F*g*(a)	1 ∀E
3	(∃z)F*g*(z)	2 ∃I

In the first derivation 'Fa' is the substitution instance associated with both the use of Universal Elimination *and* the use of Existential Introduction. In the terminology of previous sections, 'a' is the instantiating constant for these uses of the two rules. In the second derivation 'F*g*(a)' is the substitution instance associated with both the use of Universal Elimination and the use of Existential Introduction. However, the instantiating term in the use of Universal Elimination is '*g*(a)' (we have replaced '*y*' with '*g*(a)') whereas the instantiating term in the use of Existential Introduction is 'a', *not* '*g*(a)' (we replaced the constant 'a' with the variable 'z'). Since the individual term used to form substitution instances associated with the quantifier rules is sometimes an individual constant and sometimes a closed complex term, we will hereafter speak, with reference to substitution instances and uses of Existential Introduction and Universal Elimination, of the **instantiating term** rather than the instantiating constant.

But we will not modify Existential Elimination and Universal Introduction so as to allow substitution instances used in these rules to be formed from complex terms and so we will continue to talk, with reference to these latter rules, only of the **instantiating constant.** To understand why we will not modify Universal Introduction to allow for complex instantiating terms, consider the following attempt at a derivation:

Derive: (∀x)Ex

1	(∀x)E*d*(x)	Assumption	
2	E*d*(a)	1 ∀E	
3	(∀x)Ex	2 ∀I	**MISTAKE!**

If this were a legitimate derivation in *PDE* then the following argument would be valid in *PDE:*

$$\frac{(\forall x)Ed(x)}{(\forall x)Ex}$$

We do not want this argument to be valid in *PDE*. If our UD is the set of positive integers and we interpret 'Ex' as 'x is even' and '*d*(x)' as 'x times 2', the premise says that each positive integer is such that 2 times that integer is even, which is true. The conclusion says that each positive integer is even, which is false. The problem is in the attempted inference of line 3 from line 2. The expression '*d*(a)' cannot designate an arbitrarily selected member of the UD; rather it can refer only to a member of the UD that is the value of the function *d* for some member a of the UD. On the interpretation given previously, for example, '*d*(a)' can only refer to even numbers.

For similar reasons, we continue to require that in using Existential Elimination the instantiating term must be an individual constant, not a closed complex term. Here is a failed derivation that would be allowed if we dropped this requirement:

Derive: $(\exists x)Od(x)$

1	$(\exists x)Ox$	Assumption	
2	$\quad Od(a)$	A / ∃E	**MISTAKE!**
3	$\quad (\exists x)Od(x)$	2 ∃I	
4	$(\exists x)Od(x)$	1, 2–3 ∃E	**MISTAKE!**

To see why we do not want this derivation to go through suppose we again use the set of positive integers as our UD and interpret 'Ox' as 'x is odd' and '*d*(x)' as 'x times 2'. Then the primary assumption says that there is a positive integer that is odd, which is true. The sentence on line 4 says there is an integer that is 2 times some positive integer and that is odd, and this is false. The problem is that the assumption on line 2 contains information about the individual that is assumed to have property O—namely that it is the value of the function *d* for some member of the UD, while the existentially quantified sentence on line 1 does not contain this information. The requirement that the assumption for an Existential Elimination subderivation be a substitution instance formed from a constant guarantees that the assumption does not contain information that is absent from the existentially quantified sentence. Hence we continue to require that in using Existential Elimination the assumed substitution instance must be formed using an individual constant.

Having said that, it is important to note that while for Universal Introduction and Existential Elimination the instantiating term must be a constant, the substitution instances associated with these rules may contain complex terms. For example, the following is a correctly done derivation:

Derive: (∀y)E*d*(y)

1	(∀x)Ex	Assumption
2	E*d*(a)	1 ∀∃
3	(∀y)E*d*(y)	2 ∀I

Here ‘a’ is the instantiating constant for the use of Universal Introduction: In moving from line 2 to line 3 we replaced ‘a’ with ‘y’. But ‘*d*(a)’ is the instantiating term associated with Universal Elimination. In moving from line 1 to line 2 we replace ‘x’ with ‘*d*(a)’. So ‘E*d*(a)’ is a substitution instance of ‘(∀x) Ex’ because it is the result of replacing every occurrence of ‘x’ in ‘Ex’ with ‘*d*(a)’ and ‘E*d*(a)’ is a substitution instance of ‘(∀y)E*d*(y)’ because it is the result of replacing every occurrence of ‘y’ in ‘E*d*(y)’ with ‘a’.

And the following is an allowed use of Existential Elimination:

Derive:

1	(∃x)F*g*(x)	Assumption
2	F*g*(b)	A / ∃E
3	(∃z)Fz	2 ∃I
4	(∃z)Fz	1, 2–3 ∃E

Here ‘F*g*(b)’ is a substitution instance of ‘(∃x)F*g*(x)’ and also a substitution instance of ‘(∃z)Fz’. In its role as a substitution instance of ‘(∃x)F*g*(x)’, the instantiating term is ‘b’; in its role as a substitution instance of ‘(∃z)Fz’, ‘*g*(b)’ is the instantiating term.

Here are the quantifier rules, modified as appropriate for the system *PDE*.

Universal Elimination (∀E)

| (∀**x**)**P**
▷ | **P**(**t**/**x**)

Existential Introduction (∃I)

| **P**(**t**/**x**)
▷ | (∃**x**)**P**

where **t** is a closed term

Universal Introduction (∀I)

| **P**(**a**/**x**)
▷ | (∀**x**)**P**

provided that:
(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in (∀**x**)**P**.

Existential Elimination (∃E)

| (∃**x**)**P**
| | **P**(**a**/**x**)
| | **Q**
▷ | **Q**

provided that:
(i) **a** does not occur in an open assumption.
(ii) **a** does not occur in (∃**x**)**P**.
(iii) **a** does not occur in **Q**.

where **a** is an individual constant.

The definitions of the syntactic properties of sentences and sets of sentences in *PDE* (equivalence, validity, etc.) are all carried over from *PD,* substituting '*PDE'* for '*PD'* in each of the definitions.

In the rest of this section we will illustrate the use of the quantifier rules, as modified for *PDE,* by doing a series of derivations that establish various syntactic properties of sentences and sets of sentences of *PLE.*

ARGUMENTS

We begin by showing that the following argument is valid in *PDE.*

$(\forall x)(\forall y)(Fx \supset Gxy)$
$(\exists x)Ff(x)$
———
$(\exists x)(\exists y)Gxy$

Since the second premise is an existentially quantified sentence we will use Existential Elimination as our primary strategy:

Derive: $(\exists x)(\exists y)Gxy$

1	$(\forall x)(\forall y)(Fx \supset Gxy)$	Assumption
2	$(\exists x)Ff(x)$	Assumption
3	$\quad Ff(a)$	A / ∃E
G	$\quad (\exists x)(\exists y)Gxy$	
G	$(\exists x)(\exists y)Gxy$	2, 3–__ ∃E

Two applications of Universal Elimination produce a material conditional that has '$Ff(a)$' as its antecedent:

Derive: $(\exists x)(\exists y)Gxy$

1	$(\forall x)(\forall y)(Fx \supset Gxy)$	Assumption
2	$(\exists x)Ff(x)$	Assumption
3	$\quad Ff(a)$	A / ∃E
4	$\quad (\forall y)(Ff(a) \supset Gf(a)y)$	1 ∀E
5	$\quad Ff(a) \supset Gf(a)b$	4 ∀E
G	$\quad (\exists x)(\exists y)Gxy$	
G	$(\exists x)(\exists y)Gxy$	2, 3–__ 3E

We can derive ‘G*f*(a)b’ from lines 3 and 5 by Conditional Elimination, and then we can derive our current goal with two applications of Existential Introduction:

Derive: $(\exists x)(\exists y)Gxy$

1	$(\forall x)(\forall y)(Fx \supset Gxy)$	Assumption
2	$(\exists x)Ff(x)$	Assumption
3	$\quad Ff(a)$	A / $\exists$E
4	$\quad (\forall y)(Ff(a) \supset Gf(a)y)$	1 $\forall$E
5	$\quad Ff(a) \supset Gf(a)b$	4 $\forall$E
6	$\quad Gf(a)b$	3, 5 $\supset$E
7	$\quad (\exists y)Gf(a)y$	6 $\exists$I
8	$\quad (\exists x)(\exists y)Gxy$	7 $\exists$I
9	$(\exists x)(\exists y)Gxy$	2, 3–8 $\exists$E

Both Universal Elimination and Existential Introduction allow the associated substitution instance to be formed from a closed complex term, as we have done here (the substitution instance on line 4 of the universally quantified sentence on line 1 is formed using the complex term ‘*f*(a)’, as is the substitution instance on line 7 of the existentially quantified sentence on line 8).

We next show that the following argument is valid in *PDE:*

$a = g(b)$
$(\forall x)(Fxa \supset (\forall y)Gyx)$
$(\exists y)Fyg(b)$
____
$(\exists x)(\forall y)Gyx$

We will proceed much as in the previous example, using Existential Elimination as our primary strategy. But this example also requires the use of Identity Elimination:

Derive: $(\exists x)(\forall y)Gyx$

1	$a = g(b)$	Assumption
2	$(\forall x)(Fxa \supset (\forall y)Gyx)$	Assumption
3	$(\exists y)Fyg(b)$	Assumption
4	$\quad Fcg(b)$	A / $\exists$E
5	$\quad Fca \supset (\forall y)Gyc$	2 $\forall$E
6	$\quad Fca$	1, 4 =E
7	$\quad (\forall y)Gya$	5, 6 $\supset$E
8	$\quad (\exists x)(\forall y)Gyx$	7 $\exists$I
9	$(\exists x)(\forall y)Gyx$	3, 4–8 $\exists$E

At line 6 we replaced ‘*g*(b)’ in ‘Fc*g*(b)’ with ‘a’.

The sentence '(∀z) (∀y) (z = y ⊃ y = z)' says of each pair of things that if the first member of the pair is identical to the second, then the second is identical to the first. Our derivation will end with two uses of Universal Introduction:

Derive: (∀z) (∀y) (z = y ⊃ y = z)

G	b = c ⊃ c = b	
G	(∀y) (b = y ⊃ y = b)	__ ∀I
G	(∀z) (∀y) (z = y ⊃ y = z)	__ ∀I

It is important that we use two different constants to form the goal at the third line from the bottom. If we had picked 'b = b ⊃ b = b' as our goal we would not be able to derive '(∀y) (b = y ⊃ y = b)' by Universal Introduction, as the second restriction on that rule prohibits the instantiating term from occurring in the sentence that is derived by the rule. We will use Conditional Introduction to derive the goal 'b = c ⊃ c = b':

Derive: (∀z) (∀y) (z = y ⊃ y = z)

1	b = c	A / ⊃I
G	c = b	
G	b = c ⊃ c = b	1–__ ⊃I
G	(∀y) (b = y ⊃ y = b)	__ ∀I
G	(∀z) (∀y) (z = y ⊃ y = z)	__ ∀I

We can finish the derivation by using Identity Introduction to derive '(∀y)y = y' (or any other sentence of this form), then deriving either 'b = b' or 'c = 'c'—it doesn't matter which—by Universal Elimination and then using Identity Elimination to derive 'c = b':

Derive: (∀z) (∀y) (z = y ⊃ y = z)

1	b = c	A / ⊃I
2	(∀y)y = y	=I
3	c = c	2 ∀E
4	c = b	1, 3 =E
5	b = c ⊃ c = b	1–4 ⊃I
6	(∀y) (b = y ⊃ y = b)	5 ∀I
7	(∀z) (∀y) (z = y ⊃ y = z)	6 ∀I

Once we have 'c = c' at line 3 we can use Identity Elimination, replacing the second occurrence of 'c' in 'c = c' with 'b', based on the identity at line 1.

The sentence '(∀x)(∀y)(∀z)[(x = *f*(z) & y = *f*(z)) ⊃ x = y]' is also a theorem of *PDE*. We will work from the bottom up, anticipating three applications of Universal Introduction:

Derive: (∀x)(∀y)(∀z)[(x = *f*(z) & y = *f*(z)) ⊃ x = y]

1		
G	[(a = *f*(c) & b = *f*(c)) ⊃ a = b]	
G	(∀z)[(a = *f*(z) & b = *f*(z)) ⊃ a = b]	__ ∀I
G	(∀y)(∀z)[(a = *f*(z) & y = *f*(z)) ⊃ a = y]	__ ∀I
G	(∀x)(∀y)(∀z)[(x = *f*(z) & y = *f*(z)) ⊃ x = y]	__ ∀I

Our current goal is a material conditional, so we will try to obtain it by Conditional Introduction, assuming '(a = *f*(c) & b = *f*(c))' and deriving 'a = b'. The latter can be derived using Conjunction Elimination and Identity Elimination:

1	(a = *f*(c) & b = *f*(c))	A/ ⊃ I
2	a = *f*(c)	1 &E
3	b = *f*(c)	1 &E
4	a = b	2, 3 =E
5	[(a = *f*(c) & b = *f*(c)) ⊃ a = b]	1–4 ⊃E
6	(∀z)[(a = *f*(z) & b = *f*(z)) ⊃ a = b]	5 ∀I
7	(∀y)(∀z)[(a = *f*(z) & y = *f*(z)) ⊃ a = y]	6 ∀I
8	(∀x)(∀y)(∀z)[(x = *f*(z) & y = *f*(z)) ⊃ x = y]	7 ∀I

INCONSISTENCY

The set {(∀x)(Fx ∨ (∃y)Gxy), ~ F*g*(a,b), *g*(a,b) = c, ~ (∃y)Gcy} is inconsistent in *PDE*. To show this we need to derive a sentence **Q** and its negation ~ **Q**. We will use '~ F*g*(a,b)' as ~ **Q** and we will use Disjunction Elimination as our primary strategy:

Derive: F*g*(a,b), ~ F*g*(a,b)

1	(∀x)(Fx ∨ (∃y)Gxy)	Assumption
2	~ F*g*(a,b)	Assumption
3	*g*(a,b) = c	Assumption
4	~ (∃y)Gcy	Assumption
5	Fc ∨ (∃y)Gcy	1 ∀E
6	Fc	A / ∨E
7	F*g*(a,b)	3, 6 =E
8	(∃y)Gcy	A / ∨E
G	F*g*(a,b)	
G	F*g*(a,b)	5, 6–7, 8–__ ∨E
	~ F*g*(a,b)	2 R

Our remaining task is to derive ‘Fg(a,b)’. Doing so is not difficult because both ‘$\sim (\exists y)Gcy$’ and ‘$(\exists y)Gcy$’ are available to us, at lines 4 and 8, respectively. So we will use Negation Elimination to complete the derivation:

Derive: $Fg(a,b)$, $\sim Fg(a,b)$

1	$(\forall y)(Fx \vee (\exists y)Gxy)$	Assumption
2	$\sim Fg(a,b)$	Assumption
3	$g(a,b) = c$	Assumption
4	$\sim (\exists y)Gcy$	Assumption
5	$Fc \vee (\exists y)Gcy$	1 $\forall$E
6	Fc	A / $\vee$E
7	$Fg(a,b)$	3, 6 =E
8	$(\exists y)Gcy$	A / $\vee$E
9	$\sim Fg(a,b)$	A / $\sim$ E
10	$(\exists y)Gcy$	8 R
11	$\sim (\exists y)Gcy$	4 R
12	$Fg(a,b)$	9–11 $\sim$ E
13	$Fg(a,b)$	5, 6–7, 8–12 $\vee$E
14	$\sim Fg(a,b)$	2 R

There is an important difference between *PD*+ and our latest system, *PDE*. Although both are extensions of *PD* in the sense that each adds new rules to *PD, PD*+ is not stronger than *PD*. Everything derivable in *PD*+ is derivable in *PD*. However, *PDE,* with two new identity rules and modifications of two of *PD*’s quantifier rules, allows us to derive results in *PDE* that are not derivable in *PD*. The previous examples in this section involving the identity predicate and complex terms illustrate this.

However, it should be clear that we can augment the rules of *PDE* with the additional rules of *PD*+ to form a derivation system *PDE*+ that is equivalent to *PDE*. Here is a short derivation in *PDE*+:

Derive: $\sim (\exists x)f(x) = x$

1	$(\forall x)(\forall y)(\mathrm{f}(x) = y \supset \sim f(y) = x)$	Assumption
2	$f(a) = a$	A / $\sim$ I
3	$(\forall y)(f(a) = y \supset \sim f(y) = a)$	1 $\forall$E
4	$f(a) = a \supset \sim f(a) = a)$	3 $\forall$E
5	$\sim f(a) = a$	2, 4 $\supset$E
6	$f(a) = a$	2 R
7	$\sim f(a) = a$	2–6 $\sim$ I
8	$(\forall x) \sim f(x) = x$	7 $\forall$I
9	$\sim (\exists x)f(x) = x$	8 QN

10.4E EXERCISES

1. Show that each of the following is a theorem in *PDE*.

a. $a = b \supset b = a$
*b. $(a = b \,\&\, b = c) \supset a = c$
c. $(\sim a = b \,\&\, b = c) \supset \sim a = c$
*d. $\sim a = b \equiv \sim b = a$
e. $\sim a = c \supset (\sim a = b \vee \sim b = c)$

2. Show that each of the following is valid in *PDE*.

a. $a = b \,\&\, \sim Bab$

$\sim (\forall x)Bxx$

*b. $Ge \supset d = e$
$Ge \supset He$

$Ge \supset Hd$

c. $(\forall z)[Gz \supset (\forall y)(Ky \supset Hzy)]$
$(Ki \,\&\, Gj) \,\&\, i = j$

Hii

*d. $(\exists x)(Hx \,\&\, Mx)$
$Ms \,\&\, \sim Hs$

$(\exists x)[(Hx \,\&\, Mx) \,\&\, \sim x = s]$

e. $a = b$

$Ka \vee \sim Kb$

3. Show that each of the following is a theorem in *PDE*.

a. $(\forall x)(x = x \vee \sim x = x)$
*b. $(\forall x)(\forall y)(x = x \,\&\, y = y)$
c. $(\forall x)(\forall y)(x = y \equiv y = x)$
*d. $(\forall x)(\forall y)(\forall z)[(x = y \,\&\, y = z) \supset x = z]$
e. $\sim (\exists x) \sim x = x$

4. Symbolize each of the following arguments in *PLE* and show that each argument is valid in *PDE*.

a. The number 2 is not identical to 4. The numbers 2 and 4 are both even numbers. Therefore there are at least two different even numbers.
*b. Hyde killed some innocent person. But Jekyll is Hyde. Jekyll is a doctor. Hence some doctor killed some innocent person.
c. Shakespeare didn't admire himself, but the queen admired Bacon. Thus Shakespeare isn't Bacon since Bacon admired everybody who was admired by somebody.
*d. Rebecca loves those and only those who love her. The brother of Charlie loves Rebecca. Sam is Charlie's brother. So Sam and Rebecca love each other.

e. Somebody robbed Peter and paid Paul. Peter didn't rob himself. Paul didn't pay himself. Therefore the person who robbed Peter and paid Paul was neither Peter nor Paul.

5. Which of the following illustrate mistakes in *PDE*? Explain what each mistake is.

a.

1	$(\exists x)Sx$	Assumption
2	$\quad Sg(f)$	A / $\exists$E
3	$\quad (\exists x)Sg(x)$	2 $\exists$I
4	$(\exists x)Sg(x)$	1, 2–3 $\exists$E

*b.

1	$(\exists x)Sg(x,x)$	Assumption
2	$\quad Sg(i,i)$	A / $\exists$E
3	$\quad (\exists x)Sg(i,x)$	2 $\exists$I
4	$(\exists x)Sg(i,x)$	1, 2–3 $\exists$E

c.

1	$(\exists x)Hxg(x)$	Assumption
2	$\quad Heg(e)$	A / $\exists$E
3	$\quad (\exists y)Hyg(y)$	2 $\exists$I
4	$(\exists y)Hyg(y)$	1, 2–3 $\exists$E

*d.

1	$(\forall x)Rf(x)$	Assumption
2	$Rf(a)$	1 $\forall$E
3	$(\forall z)Rf(z)$	2 $\forall$I

e.

1	$(\forall x)Lxxx$	Assumption
2	$Lf(a,a)a$	1 $\forall$E
3	$(\forall x)Lf(x,x)x$	2 $\forall$I

*f.

1	$(\forall x)Mx$	Assumption
2	$Mf(f(a))$	1 $\forall$E
3	$(\exists x)Mf(x)$	2 $\exists$I

g.

1	$(\forall x)Rf(x,x)$	Assumption
2	$Rf(c,c)$	1 $\forall$E
3	$(\forall y)Ry$	2 $\forall$I

*h.

1	$(\forall x)Jx$	Assumption
2	$Jf(f(a))$	1 $\forall$E
3	$(\exists y)Jf(f(y))$	2 $\exists$I

i.

1	$(\forall x)Jx$	Assumption
2	$Jf(g(a,b))$	1 ∀E
3	$(\exists x)Jf(g(x,b))$	2 ∃I

*j.

1	$(\forall x)Lx$	Assumption
2	$Lf(a,a)$	1 ∀E
3	$(\forall x)Lf(a,x)$	2 ∀I

6. Show that each of the following is a theorem in *PDE*.

a. $(\forall x)(\exists y)f(x) = y$

*b. $(\forall x)(\forall y)(\forall z)[(f(x) = g(x,y) \ \& \ g(x,y) = h(x,y,z)) \supset f(x) = h(x,y,z)]$

c. $(\forall x)Ff(x) \supset (\forall x)Ff(g(x))$

*d. $(\forall x)[\sim f(x) = x \supset (\forall y)(f(x) = y \supset \sim x = y)]$

e. $(\forall x)(f(f(x)) = x \supset f(f(f(f(x)))) = x)$

*f. $(\forall x)(\forall y)(\forall z)[(f(g(x)) = y \ \& \ f(y) = z) \supset f(f(g(x))) = z]$

g. $(\forall x)(\forall y)[(f(x) = y \ \& \ f(y) = x) \supset x = f(f(x))]$

7. Show that each of the following is valid in *PDE*.

a. $(\forall x)(Bx \supset Gxf(x))$
$(\forall x)Bf(x)$
———
$(\forall x)Gf(x)f(f(x))$

*b. $(\forall x)(Kx \vee Hg(x))$
———
$(\forall x)(Kg(x) \vee Hg(g(x)))$

c. $(\forall x)(\forall y)(f(x) = y \supset Myxc)$
$\sim Mbac \ \& \sim Mabc$
———
$\sim f(a) = b$

*d. $\sim (\exists x)Rx$
———
$(\forall x) \sim Rf(x,g(x))$

e. $(\exists x)(\forall y)(\forall z)Lxyz$
———
$(\exists x)Lxf(x)g(x)$

*f. $(\forall x)[\sim Lxf(x) \vee (\exists y)Ng(y)]$
———
$(\exists x)Lf(x)f(f(x)) \supset (\exists x)Ng(y)$

g. $(\forall x)[Zx \supset (\forall y)(\sim Dxy \equiv Hf(f(y)))]$
$(\forall x)(Zx \ \& \sim Hx)$
———
$(\forall x)Df(x)f(x)$

*h. $(\forall x)(\forall y)(\exists z)Sf(x)yz$
$(\forall x)(\forall y)(\forall z)(Sxyz \supset \sim (Cxyz \vee Mzyx))$
———
$(\exists x)(\exists y) \sim (\forall z)Mzg(y)f(g(x))$

DERIVABILITY IN *PD:* A sentence **P** of *PL* is *derivable in PD* from a set Γ of sentences of *PL* if and only if there is a derivation in *PD* in which all the primary assumptions are members of Γ and **P** occurs within the scope of only those assumptions.
VALIDITY IN *PD:* An argument of *PL* is *valid in PD* if and only if the conclusion of the argument is derivable in *PD* from the set consisting of the premises. An argument of *PL* is *invalid in PD* if and only if it is not valid in *PD*.
THEOREM IN *PD:* A sentence **P** of *PL* is a *theorem in PD* if and only if **P** is derivable in *PD* from the empty set.
EQUIVALENCE IN *PD:* Sentences **P** and **Q** of *PL* are *equivalent in PD* if and only if **Q** is derivable in *PD* from {**P**} and **P** is derivable in *PD* from {**Q**}.
INCONSISTENCY IN *PD:* A set Γ of sentences of *PL* is *inconsistent in PD* if and only if there is a sentence **P** of *PL* such that both **P** and ~ **P** are derivable in *PD* from Γ. A set Γ of sentences of *PL* is *consistent in PD* if and only if it is not inconsistent in *PD*.

[2]Similar definitions hold for the derivation systems *PD+*, *PDE,* and *PDE+*.