LOGIC:
A COMPUTER APPROACH

Morton L. Schagrin
State University of New York at Fredonia

William J. Rapaport
State University of New York at Buffalo

Randall R. Dipert
State University of New York at Fredonia

Ch 13 - Predicate Logic:
Quantifier Inference Rules

© 1985
McGraw-Hill Book Company
New York St. Louis San Francisco Auckland Bogotá Hamburg Johannesburg London Madrid Mexico Montreal New Delhi Panama Paris São Paulo Singapore Sydney Tokyo Toronto

are "named" by numbers, they can carry their dossiers with them.) We next need to produce all the minimal models. This task amounts to producing all the various combinations of representative individuals.

Once the minimal models have been produced, we need to devise a procedure for determining the truth values of the premises and conclusion in a given model. As we have seen, this can be accomplished by borrowing the main technique from WANG'S ALGORITHM (step 3 in Chapter 6) and by adding the conditions whose "main connective" is in fact a quantifier. For suggestions on programming these steps, consult the implementation suggestions at the end of Chapter 6. Note that we also need a slightly different TEST procedure than that given in Chapter 6 for WANG'S ALGORITHM.

Exercises

1. Write an algorithm (or program) that inputs the sentences of an argument and determines the total number of distinct predicates in them.
2. Write an algorithm (or program) that takes a list of distinct predicates (either as input or produced by another procedure in the same program) and generates all the representative individuals.
3. Modify the algorithm MAIN-CONNECTIVE in Chapter 6 so that if a formula is universally or existentially quantified, the quantifier whose scope covers the rest of the formula is identified as the "main connective" of the formula. (The initial quantifier is not really a connective, since it does not connect sentences. It is a "connective" much like negation is a connective.)
4. Write an algorithm (or program) that takes as input a universally or existentially quantified formula and a list of individuals and then does the following: (a) deletes the initial quantifier and (b) outputs all the *instances* of the resulting formula.
5. Write an algorithm (or program) to determine when a formula is a simple predicate formula—a formula containing only a predicate and individual constants.

PREDICATE LOGIC: Quantifier Inference Rules

A valid argument, we recall, is an argument where it is impossible for the premises to be TRUE and the conclusion FALSE. In the logic of sentences, there are several ways to determine if an argument is valid or invalid. We can, in the logic of sentences, construct a truth table and examine every situation to see if it is possible for the premises to be TRUE and the conclusion FALSE. In predicate logic, however, it is not possible, in general, to construct or to inspect all models, including those models in which the premises are TRUE or the conclusion FALSE. Consequently, other ways of showing an argument to be valid must be found. One of the simplest ways is to derive the conclusion from the premises in a formal deduction system with truth-preserving rules of inference. (The system should also be complete, in the sense that every conclusion of a valid argument can be derived in the system.)

We turn now to the problem of deriving conclusions with sentences containing quantifiers and variables. The whole point of symbolizing the structure of sentences with quantifiers and variables is to enable ourselves to derive conclusions that we could not prove by the methods of sentential logic alone.

We feel sure that

1. All Greeks are mortal.
2. Socrates is a Greek.

∴ 3. Socrates is mortal.

is a valid argument. We need to see why it is valid, and also we need to develop rules that will allow us to derive the conclusion from the premises.

When the above argument is symbolized, we get something like

1. ∀*x*(G*x* → M*x*)
2. G*c*

∴ 3. M*c*

Sentence (1) "says" that for every individual *x*, '(G*x* → M*x*)' is satisfied by that individual. Hence, an instance using '*c*', the name of Socrates, for '*x*' is TRUE. That is, if (1) has the truth value TRUE, so does

1a. (G*c* → M*c*)

But now we can use →ELIM on (1a) and (2) to obtain (3).

The earlier rules for introducing and eliminating connectives in a sentential derivation remain unchanged. We need only to add some rules for introducing and eliminating quantifiers. Our general strategy will be to eliminate quantifiers somehow, manipulate and transform the results using the earlier sentential rules, and, finally, introduce appropriate quantifiers, if needed, to obtain the desired conclusion. These new rules for quantifier INTRO and ELIM are very precisely stated, and careful attention must be paid not only to the sentence on the line to which the rule is applied but also to other sentences in the proof or subproof.

Universal Quantifier Rules

Our earlier rules of inference from Chapters 8 and 9 apply to quantified sentences considered as atomic sentences. Thus &ELIM will apply to a line with the sentence '(∀*x*F*x* & ∀*y*G*y*)' on it. We take '∀*x*F*x*' as a single sentence **P** and '∀*y*G*y*' as **Q.** That is, we take '(∀*x*F*x* & ∀*y*G*y*)' as having the form (**P** & **Q**). Now, however, we are going to extend our deduction system in order to make additional derivations to and from quantified sentences.

Universal Elimination

Reflecting on the truth conditions for a universally quantified sentence, say '∀*x*F*x*', we note that it has the value TRUE only if all instances of 'F*x*' also have the value TRUE. Hence, if we infer an instance, any instance, from '∀*x*F*x*', we shall never move from a true sentence to a false one. This provides a justification for the rule:

∀ELIM RULE: From a sentence of the form
∀vP
you may derive
P[c/v]

In the statement of the rule, '**v**' is used for any variable at all (*w, x, y, z,* . . .) and '**c**' for any constant at all (*a, b, c, d,* . . .). We use the notation **P[c/v]** for the result of replacing all free occurrences of the variable **v** in formula **P** with the constant **c.**

For example, if **P** is '∃*x*(F*x* v G*y*)', then **P**[*a*/*y*] is '∃*x*(F*x* v G*a*)', but **P**[*a*/*x*] is still '∃*x*(F*x* v G*y*)', since '*x*' is not free in **P.**

In a derivation, the use of ∀ELIM would look like this:

10. ∀*x*(F*x* → H*x*) :<PREMISE or Rule>
.
.
.
15. (F*d* → H*d*) :∀ELIM,10

The sentence on line 10 is universally quantified, and the sentence on line 15 results from the one on line 10 by deleting the initial quantifier and replacing all now free occurrences of the quantifier variable '*x*' with the individual constant '*d*'.

It is essential to note that this rule and all the other quantifier rules require the scope of the initial quantifier to stretch to the end of the sentence on that line. Here is an example of a sentence where ∀ELIM cannot be used, because the scope of the universal quantifier expression '∀*x*' is not the entire sentence:

(∀*x*F*x* v ∀*y*G*y*)

From this sentence, one cannot get '(F*a* v ∀*y*G*y*)' by ∀ELIM.

One can use ∀ELIM several times over, citing the same line:

10. ∀*x*(F*x* → H*x*)
.
.
.
15. (F*d* → H*d*) :∀ELIM,10
16. (F*e* → H*e*) :∀ELIM,10

Universal Introduction

Next, we would like to be able to generalize, that is, to introduce a universal quantifier. A clue to justifying this move can be found in elementary geometry classes, where the teacher draws a triangle on the board and then uses this specific triangle to prove theorems about *all* triangles. This works as long as no appeal is made to any special properties of the example triangle. That is, if we can prove something about an *arbitrarily* selected individual, we have proved it for *any* individual. We need only to ensure that special properties of the selected individual play no role in the proof. The following rule is qualified to ensure just that.

∀INTRO RULE: From a sentence
P
you may derive
∀**vP[v/c]**
Provided that:
1. **c** does not occur in any premise.
2. If **P** occurs in a subproof, no constant in **P** occurs in an ASSUMPTION still in force.
3. All new occurrences of the variable **v** in **P** are free after the replacement in **P[v/c]**.

In proviso (2), an ASSUMPTION is "still in force" during the subproof following it and during any sub-subproofs within that. Finally, the notation **P[v/c]** means that all occurrences of the constant **c** are replaced by the variable **v.** So, proviso (3) means that when **v** replaces **c,** it should not fall within the scope of a quantifier already present that uses **v.** The new occurrence of the variable should not, so to speak, be "captured" by a quantifier already present in the wff **P.**

Examples of the correct use of ∀INTRO are given below. Assume throughout that the restrictions on the constant on line 5 are all met.

Example 1.
5. (F*a* → G*a*) :<Rule>
.
.
.
9. ∀*x*(F*x* → G*x*) :∀INTRO,5

Example 2.
5. (F*b* v ∃*y*G*y*) :<Rule>
.
.
.
12. ∀*x*(F*x* v ∃*y*G*y*) :∀INTRO,5

Here is a complete derivation for the following argument:

∀*x*(F*x* → G*x*)
∴ (∀*y*F*y* → ∀*z*G*z*)

1. ∀*x*(F*x* → G*x*) :PREMISE
/BEGIN: (∀*y*F*y* → ∀*z*G*z*) by →INTRO/
*2. ∀*y*F*y* :ASSUMPTION
*3. F*a* :∀ELIM,2
*4. ∀*x*(F*x* → G*x*) :SEND,1
*5. (F*a* → G*a*) :∀ELIM,4
*6. G*a* :→ELIM,5,3
*7. ∀*z*G*z* :∀INTRO,6
*8. (∀*y*F*y* → ∀*z*G*z*) :→INTRO,2,7
/END: (∀*y*F*y* → ∀*z*G*z*)/
9. (∀*y*F*y* → ∀*z*G*z*) :RETURN,8

Here is an incorrect use of ∀INTRO:

Example 3.
5. (F*a* → G*a*) :<Rule>
.
.
.
9. ∀*x*(F*x* → G*a*) :∀INTRO,5 [INCORRECT—Not all occurrences of '*a*' replaced.]

Another incorrect use of ∀INTRO is:

Example 4.
5. (F*a* → ∃*x*(G*a* & H*x*)) :<Rule>
.
.
.
9. ∀*x*(F*x* → ∃*x*(G*x* & H*x*)) :∀INTRO,5 [INCORRECT—the '*x*' in 'G*x*' was captured.]

Observe that the replacement of '*a*' in 'G*a*' by '*x*' in line 5 led to its being captured by the existential quantifier already there. Instead of '*x*', we could use another variable, say '*y*', and correctly infer:

9. ∀*y*(F*y* → ∃*x*(G*y* & H*x*)) :∀INTRO,5

With these two rules we can derive the conclusions of some arguments traditionally studied since the time of Aristotle. One, for instance, is the ancient syllogistic argument:

All humans are mortal.
All Greeks are humans.
∴ All Greeks are mortal.

The first step, of course, is to symbolize the English sentences:

∀*x*(H*x* → M*x*)
∀*x*(G*x* → H*x*)
∴ ∀*x*(G*x* → M*x*)

A proof of the conclusion using our two quantification rules goes as follows:

1. ∀*x*(H*x* → M*x*) :PREMISE
2. ∀*x*(G*x* → H*x*) :PREMISE
3. (H*a* → M*a*) :∀ELIM,1
4. (G*a* → H*a*) :∀ELIM,2
5. (G*a* → M*a*) :HS,2,1
6. ∀*x*(G*x* → M*x*) :∀INTRO,5

The restrictions on the rule ∀INTRO prevent the following attempted derivation:

1. ∀*x*(G*x* → M*x*) :PREMISE
2. G*f* :PREMISE
3. (G*f* → M*f*) :∀ELIM,1
4. M*f* :→ELIM,2,3
5. ∀*x*M*x* :∀INTRO,4 [INCORRECT]

Here, '*f*' occurs in PREMISE 2 and cannot be generalized on.

Existential Quantifier Rules

Having a pair of rules for introducing and eliminating universal quantifiers, we need now to develop a pair of rules of inference to introduce and eliminate existential quantifiers.

Existential Introduction

The next rule is again easy to justify. If something is true of a particular individual, then there is some individual for which it is true. Schematically,

F*a*
∴ ∃*x*F*x*

The correct statement of the rule is:

∃INTRO From a sentence of the form
  P[c/v]
 you may derive
  ∃vP

In this rule, the constant **c** replaces all *free* occurrences of the variable **v** in well-formed formula **P.**

You may find the statement of this rule to be odd, because as you move down the lines of a derivation, you encounter the sentence **P[c/v]** before you come to the sentence with the variable **v,** namely, **∃vP.** But to use the rule correctly, you need only to ensure that the earlier sentence and the wff you are about to existentially quantify are properly related: The earlier one can be obtained from **P** by replacing all free occurrences of **v** with **c.** In addition, if you are following a modified version of PROOF-GIVER, then you will, in fact, encounter **∃vP** first in your task file before you get to **P[c/v]**. This is because the task file begins at the end of the derivation and works up to the premises.

These are all correct uses of the rule ∃INTRO:

n.	F*aa*	F*aa*	F*aa*	F*aa*	:<PREMISE or Rule>
	.	.	.	.	
	.	.	.	.	
	.	.	.	.	
n + *k*.	∃*x*F*xx*	∃*x*F*xa*	∃*x*F*ax*	∃*x*F*aa*	:∃INTRO,*n*

Each of these is a truth-preserving inference allowed by the rule. Any argument with line *n* as premise and line (*n* + *k*) as conclusion is a valid argument. Each of the four simple inferences above is allowable by the rule ∃INTRO. Moreover, in our previous example, although we could not derive '∀*x*M*x*' ('Everything is mortal'), we could at line 5 derive '∃*x*M*x*' ('Something is mortal').

1. ∀*x*(G*x* → M*x*) :PREMISE
2. G*f* :PREMISE
3. (G*f* → M*f*) :∀ELIM,1
4. M*f* :→ELIM,3,2
5. ∃*x*M*x* :∃INTRO,4

Existential Elimination

The final rule, ∃ELIM, deals with the sorts of inferences one can validly make from an existentially quantified sentence. Here we take a cue from legal practice. Frequently, in legal situations, we know that *someone* committed the crime, but we don't know who specifically it was. A warrant is issued for someone, John Doe or Jane Doe. We then reason about, say, John Doe, although we don't know exactly who he is. Whatever conclusion we reach that does not refer to John Doe *by that name* is, in general, a correct conclusion.

Our strategy with an existentially quantified sentence is to name someone as John Doe and see what follows. If we reach a conclusion that does not depend on someone's actually being named John Doe, then that is a valid conclusion from the original statement referring, nonspecifically, to someone or other. Let us look at the rule and practice using it.

∃ELIM   If a sentence on a previous line has the form
  ∃vP
and there is a subproof beginning with ASSUMPTION
  P[c/v]
where constant **c** is new to the proof, and ending with a sentence
  Q
not containing **c,**
then **Q** may be RETURNed from that subproof.

To say that a constant is "new to the proof" means, simply, that it has not been used before. Note that the RETURN rule has now been slightly, but significantly, expanded.

Rule ∃ELIM is different from any of the other elimination rules because it is not a rule for eliminating an existential quantifier from a line. It is more like a strategy for constructing subproofs to derive conclusions from existentially quantified sentences.

Let us work a few examples, again drawn from traditional Aristotelian logic.

All circus animals are tame animals.
Some lions are circus animals.
∴ Some lions are tame animals.

1. ∀*x*(C*x* → A*x*)   :PREMISE
2. ∃*x*(L*x* & C*x*)   :PREMISE
  /BEGIN: ∃ELIM/
*3. (L*a* & C*a*)   :ASSUMPTION for ∃ELIM,2
*4. ∀*x*(C*x* → A*x*)   :SEND,1
*5. (C*a* → A*a*)   :∀ELIM,4
*6. C*a*   :&ELIM,3
*7. A*a*   :→ELIM,5,6
*8. L*a*   :&ELIM,3
*9. (L*a* & A*a*)   :&INTRO,8,7
*10. ∃*x*(L*x* & A*x*)   :∃INTRO,9
  /END: ∃ELIM/
11. ∃x(L*x* & A*x*)   :RETURN,10

In line 3, we assumed that *a* is a lion who is a circus animal. The subproof concludes on line 10 with a sentence that does not mention *a* and thus does not depend on the

assumption that '*a*' is a name of a circus lion. So, the information on line 10 may be returned to the main proof.*

Some Examples

When using the quantifier introduction and elimination rules, one must take care that the scope of the quantifier (introduced or eliminated) is the entire sentence on the line. We shall examine some ways of dealing with sentences having quantifiers whose scope is only a proper part of the sentence. For instance, in

(∀*x*F*x* & A)

the scope of '∀*x*' is just the left conjunct. The inner structure of the right conjunct, 'A', is of no concern here; it can be any sentence whatever, with one caution to be explained shortly. We cannot use ∀ELIM on this sentence as it stands, but we can derive another sentence from it to which ∀ELIM can apply:

1. (∀*x*F*x* & A)   :PREMISE
2. ∀*x*F*x*   :&ELIM,1
3. A   :&ELIM,1
4. F*a*   :∀ELIM,2
5. (F*a* & A)   :&INTRO,4,3
6. ∀*x*(F*x* & A)   :∀INTRO,5

The derivation assumes that the sentence A does not contain the constant '*a*'. If there are constants in sentence A, then the constant introduced at line 4 should be different from any of them.

Here is another simple derivation that moves a quantifier to the beginning of the sentence:

1. (A → ∀*x*F*x*)   :PREMISE
  /BEGIN: → INTRO for (A → F*a*)/

**Historical note:* The argument above was stated as a correct Aristotelian syllogistic argument. The noun phrase "tame animals" must be used, although to a modern ear, the sentence sounds stilted. With our symbolism, we could deal directly with the more naturally sounding argument:

All circus animals are tame.
Some lions are circus animals.
∴ Some lions are tame.

The symbolic form remains the same. The difference is that earlier, 'A*x*' symbolized '*x* is a tame animal', while now it symbolizes '*x* is tame'.

*2. A :ASSUMPTION
*3. (A → ∀xFx) :SEND,1
*4. ∀xFx :→ELIM,3,2
*5. Fa :∀ELIM,4
*6. (A → Fa) :→INTRO,2,5
/END: →INTRO/

7. (A → Fa) :RETURN,6
8. ∀x(A → Fx) :∀INTRO,7

Quantifier Negation Rule

Before we work on the next examples, it will be helpful to consider the cases where a negation sign precedes a quantifier whose scope is the rest of the sentence. There are two kinds of cases:

~∀xFx ~∃xFx

We propose to show that

'~∃xFx' is logically equivalent to '∀x~Fx'.

The equivalence between the other two, '~∀xFx' and '∃x~Fx', is shown similarly; it is an exercise at the end of the chapter. The results are of some importance, since these logical equivalences open the way to using the rule of replacement on wffs with quantifiers flanked by negation signs.

One way to show the equivalence is by way of a *semantic* discussion of the truth conditions for the pair of sentences. Thus we would begin by pointing out that

'~∃xFx' is TRUE if and only if '∃xFx' is FALSE

and that

'∃xFx' is FALSE if and only if every instance of 'Fx' is FALSE.

But this is so if and only if every instance of '~Fx' is TRUE, and that is the condition if and only if '∀x~Fx' is TRUE.

A second way of showing equivalence is to prove that the biconditional of the two is a theorem. Thus we shall prove '(~∃xFx ↔ ∀x~Fx)' beginning with no premises. There is a problem of strategy during the derivation, and we shall interrupt the derivation at that point to discuss the problem and a solution.

*1. ~∃xFx :ASSUMPTION
/BEGIN: ~INTRO to derive ~Fa/
**2. Fa :ASSUMPTION
**3. ∃xFx :∃INTRO,2
**4. ~∃xFx :SEND,1
**5. ~Fa :~INTRO,2,3,4
/END: ~INTRO/
*6. ~Fa :RETURN,5
*7. ∀x~Fx :∀INTRO,6
*8. (~∃xFx → ∀x~Fx) :→INTRO,1,7
/END: →INTRO/
9. (~∃xFx → ∀x~Fx) :RETURN,8
*10. ∀X~Fx :ASSUMPTION
/BEGIN: ~INTRO to derive ~∃xFx/
**11. ∃xFx :ASSUMPTION
/BEGIN: ∃ELIM/
***12. Fb :ASSUMPTION for ∃ELIM,2
***13. ∀x~Fx :SEND,10
***14. ~Fb :∀ELIM,13

We now have a problem: A contradiction can be seen on lines 12 and 14, but since they contain '*b*'—the constant in the assumption at 12—they cannot be returned out of the subproof. But given a contradiction, any sentence can be proved—in particular, a contradiction without the constant '*b*'. Letting 'A' be an atomic sentence (say, 'Grass is green'), we continue the proof.

***15. (Fb v (A & ~A)) :vINTRO,12
***16. (A & ~A) :vELIM,15,14
/END: ∃ELIM/
**17. (A & ~A) :RETURN,16
**18. A :&ELIM,17
**19. ~A :&ELIM,17
**20. ~∃xFx :~INTRO,11,18,19
/END: ~INTRO/
*21. ~∃xFx :RETURN,20
*22. (∀x~Fx → ~∃xFx) :→INTRO,10,21
/END: →INTRO/
23. (∀x~Fx → ~∃xFx) :RETURN,22
24. (~∃xFx ↔ ∀x~Fx) :↔INTRO,9,23

Since 'Fx' played no significant role in the above proof, this result holds for any wff in place of 'Fx'. We can now adopt a quantifier negation rule:

QUANTIFIER NEGATION (QN) ~∃vS is derivable iff ∀v~S is derivable.
~∀vS is derivable iff ∃v~S is derivable.

This rule enables us, at any time in a proof, to "move the negation sign through a quantifier" if we change the *quantity* of the quantifier. This derived rule is very useful, as the following proof shows:

To prove that '∀x(Fx → A)' is logically equivalent to '(∃xFx → A)'

A proof, using no premises, of the biconditional:

*1.	∀x(Fx → A)	:ASSUMPTION
	/BEGIN: → INTRO for (∃xFx → A)/	
**2.	∃xFx	:ASSUMPTION
	/BEGIN: ∃ELIM/	
***3.	Fa	:ASSUMPTION for ∃ELIM,2
***4.	∀x(Fx → A)	:SEND,1
***5.	(Fa → A)	:∀ELIM,4
***6.	A	: → ELIM,3,5
	/END: ∃ELIM/	
**7.	A	:RETURN,6
**8.	(∃xFx → A)	: → INTRO,2,7
	/END: → INTRO/	
*9.	(∃xFx → A)	:RETURN,8
*10.	(∀x(Fx → A) → (∃xFx → A))	: → INTRO,1,9
	/END: → INTRO/	
11.	(∀x(Fx → A) → (∃xFx → A))	:RETURN,10
*12.	(∃xFx → A)	:ASSUMPTION
	/BEGIN: ~ELIM to derive ∀x(Fx → A)/	
**13.	~∀x(Fx → A)	:ASSUMPTION
**14.	∃x ~(Fx → A)	:QN,13
	/BEGIN: ∃ELIM/	
***15.	~(Fb → A)	:ASSUMPTION for ∃ELIM,14
***16.	~~(Fb & ~A)	:RR EQ,15
***17.	(Fb & ~A)	:RR DN,16
***18.	Fb	:&ELIM,17
***19.	∃xFx	:∃INTRO,18
***20.	~A	:&ELIM,17
***21.	(∃xFx & ~A)	:&INTRO,20,19
	/END: ∃ELIM/	
**22.	(∃xFx & ~A)	:RETURN,21
**23.	∃xFx	:&ELIM,22
**24.	(∃xFx → A)	:SEND,12
**25.	A	: → ELIM,24,23

**26.	~A	:&ELIM,22
**27.	∀x(Fx → A)	:~ELIM,13,25,26
	/END: ~ELIM/	
*28.	∀x(Fx → A)	:RETURN,27
*29.	((∃xFx → A) → ∀x(Fx → A))	: → INTRO,12,28
	/END: → INTRO/	
30.	((∃xFx → A) → ∀x(Fx → A))	:RETURN,29
31.	(∀x(Fx → A) ↔ (∃xFx → A))	:↔INTRO,11,30

Let us apply our expanded set of rules to a few examples in order to become more familiar with proofs. The first example has some historical interest. The British logician and logic-machine builder W. S. Jevons, modifying an example from Augustus De Morgan, accused traditional Aristotelian logic of being unable to validate this argument:

Horses are animals.
Therefore, every head of a horse is a head of an animal.

Using '*Dyx*' for '*y* is a head of *x*', we can symbolize these sentences as:

∀x(Hx → Ax)
∴∀y(∃x(Hx & Dyx) → ∃z(Az & Dyz))

Now working back from the conclusion, we can devise a simple proof.

1.	∀x(Hx → Ax)	:PREMISE
	/BEGIN: → INTRO for (∃x(Hx & Dax) → ∃z(Az & Daz))/	
*2.	∃x(Hx & Dax)	:ASSUMPTION
	/BEGIN: ∃z(Az & Daz)/	
**3.	(Hb & Dab)	:ASSUMPTION for ∃ELIM,2
**4.	∀x(Hx → Ax)	:SEND,1
**5.	(Hb → Ab)	:∀ELIM,4
**6.	Hb	: & ELIM,3
**7.	Ab	: → ELIM,6,5
**8.	Dab	:&ELIM,3
**9.	(Ab & Dab)	: & INTRO,7,8
**10.	∃z(Az & Daz)	:∃INTRO,9
	/END: ∃ELIM/	
*11.	∃z(Az & Daz)	:RETURN,10
*12.	(∃x(Hx & Dax) → ∃z(Az & Daz))	: → INTRO,2,11
	/END: → INTRO/	
13.	(∃x(Hx & Dax) → ∃z(Az & Daz))	:RETURN,12
14.	∀y(∃x(Hx & Dyx) → ∃z(Az & Dyz))	:∀INTRO,13

This concludes the proof. Notice that line 13 contains '*a*' with no restrictions on ∀INTRO, since the assumptions at lines 2 and 3 are no longer in force.

The next example illustrates how we can handle the identity relation with our present notation. [We could, by the way, extend our present system to treat the identity relation in a special way, with special rules of inference for formulas with an identity sign (=).] Consider this argument:

If one event causes another event, the first event begins before the second. When one event begins before another, the events are not identical. Every event is identical to itself. Hence, no event is its own cause.

Our dictionary for symbolizing is:

Cxy: x causes y
Bxy: x begins before y
Ixy: x is identical to y

We symbolize this argument as follows:

1. ∀x∀y(Cxy → Bxy)
2. ∀x∀y(Bxy → ~Ixy)
3. ∀xIxx
∴ 4. ∀x~Cxx

Observe that the conclusion is a universally quantified sentence. This suggests that in the last step in the derivation, the rule ∀INTRO is applied. As usual, our strategy will be to eliminate quantifiers first, perform sentence transformations, and then introduce quantifiers where needed.

1.	∀x∀y(Cxy → Bxy)	:PREMISE
2.	∀x∀y(Bxy → ~Ixy)	:PREMISE
3.	∀xIxx	:PREMISE
4.	Iaa	:∀ELIM,3
5.	∀y(Bay → ~Iay)	:∀ELIM,2
6.	(Baa → ~Iaa)	:∀ELIM,5
7.	∀y(Cay → Bay)	:∀ELIM,1
8.	(Caa → Baa)	:∀ELIM,7
9.	~~Iaa	:RR DN,4
10.	~Baa	:MT,6,9
11.	~Caa	:MT,8,10
12.	∀x~Cxx	:∀INTRO,11

Our final example will give us some practice with the quantifier negation rule:

Not all successful people are rich. But all successful people are either happy or rich. So, there are some people who are not rich and yet who are happy.

Symbolizing this with some care, we get:

1. ~∀x(Sx → Rx)
2. ∀x(Sx → (Rx v Hx))
∴ 3. ∃x(~Rx & Hx)

One proof of this argument is:

1.	~∀x(Sx → Rx)	:PREMISE
2.	∀x(Sx → (Rx v Hx))	:PREMISE
3.	∃x~(Sx → Rx)	:QN,1
	/BEGIN: ∃ELIM to derive ∃x(~Rx & Hx)/	
*4.	~(Sa → Ra)	:ASSUMPTION for ∃ELIM,3
*5.	∀x(Sx → (Rx v Hx))	:SEND,2
*6.	(Sa → (Ra v Ha))	:∀ELIM,5
*7.	~~(Sa & ~Ra)	:RR EQ,4
*8.	(Sa & ~Ra)	:RR DN,7
*9.	Sa	:&ELIM,8
*10.	(Ra v Ha)	:→ELIM,9,6
*11.	~Ra	:&ELIM,8
*12.	Ha	:vELIM,11,10
*13.	(~Ra & Ha)	:&INTRO,11,12
*14.	∃x(~Rx & Hx)	:∃INTRO,13
	/END: ∃ELIM/	
15.	∃x(~Rx & Hx)	:RETURN,14

This concludes the proof. It will be very helpful for you to review these examples and to work some of the related exercises at the end of the chapter.

Invalid Arguments

We have been deriving conclusions of valid arguments. But what if an argument is invalid? How would we show that an argument is invalid? Consider this argument:

All circus animals are tame.
Some lions are not circus animals.
∴ Some lions are not tame.

∀x(Cx → Ax)
∃x(Lx & ~Cx)
∴ ∃x(Lx & ~Ax)

Try as we might, we would not be able to produce the indicated conclusion using our rules. And it is well that we cannot, for the conclusion is not a logical consequence of the premises. But how do we show that it is not?

To show an argument to be *invalid,* we must provide a model in which the premises are true sentences but the conclusion is a false one. That is, we must describe a model where inspection of the dossiers on individuals in the model reveals that the premises are TRUE but the conclusion is FALSE. There are many such models for the argument we are now considering; here is one with just two individuals:

	C	A	L
a	1	1	0
b	0	1	1

We can see that both '(C*a* → A*a*)' and '(C*b* → A*b*)' are TRUE in this model. Thus

'∀*x*(C*x* → A*x*)' is TRUE in the model.

Furthermore, V(L*b* & ~C*b*) = TRUE, so

'∃*x*(L*x* & ~C*b*)' is TRUE in the model.

But V(L*a* & ~A*a*) = FALSE, and V(L*b* & ~A*b*) = FALSE also. Since there are no other individuals,

'∃*x*(L*x* & ~A*x*)' is FALSE in the model.

There is no algorithm for finding models that invalidate an argument. However, some procedures and rules of thumb can be devised for this search task, as we shall see in the next chapter.

Summary

Two universal quantification rules were discussed: universal elimination (∀ELIM) and universal introduction (∀INTRO).

∀ELIM—From a sentence of the form **∀vP,** you may derive **P[c/v]**.
∀INTRO—From a sentence **P,** you may derive **∀vP[v/c]**, *provided that:*
1. **c** does not occur in any premise.
2. If **P** is in a subproof, no constant in **P** occurs in an ASSUMPTION still in force.
3. All new occurrences of **v** in **P** are free after the replacement in **P[v/c]**.

The notation **P[c/v]** means that the constant **c** replaces all free occurrences of the variable **v** in **P.** Similarly, **P[v/c]** means that the variable **v** replaces all occurrences of the constant **c** in **P** and is free after replacement.

Two existential quantification rules were also discussed: existential introduction (∃INTRO) and existential elimination (∃ELIM).

∃INTRO—From a sentence **P[c/v]**, you may derive **∃vP.**
∃ELIM—If a sentence has the form **∃vP,** and there is a subproof with ASSUMPTION **P[c/v]**, where **c** is new to the whole proof, and the subproof ends with sentence **Q** not containing **c,** then **Q** may be RETURNed from the subproof.

A quantifier negation rule was (partially) proved:

~**∃vS** is derivable iff **∀v~S** is derivable.
~**∀vS** is derivable iff **∃v~S** is derivable.

This rule enables us to move negation signs back and forth through quantifiers, if we change the quantity of the quantifiers.

Some examples were worked, and then the problem of showing an argument to be invalid was introduced.

Exercises

A. Construct derivations for the following arguments:

1. ∀*x*(F*x* → G*x*)
   ∃*x*(F*x* & H*x*)
   ∴∃*x*(G*x* & H*x*)
2. ~∀*x*(F*x* → G*x*)
   ∴∃*x*(F*x* & ~G*x*)
3. ~∃*x*(F*x* & ~G*x*)
   ∴∀*x*(F*x* → G*x*)
4. ∀*x*(F*x* → ∃*y*R*xy*)
   ∀*x*∀*y*(R*xy* → G*x*)
   ∃*x*F*x*
   ∴∃*x*G*x*
5. ∀*x*(F*x* → G*x*)
   (∃*x*G*x* → ∃*x*(H*x* & D*x*))
   ∴(∃*x*F*x* → ∃*x*H*x*)
6. ~∃*x*F*x*
   ∴∀*x*(F*x* → G*x*)
7. ~∀*x*(F*x* → ~G*x*)
   ∴∃*x*(F*x* & G*x*)
8. ~∃*x*(F*x* & G*x*)
   ∴∀*x*(F*x* → ~G*x*)
9. ∀*x*~G*x*
   ∀*x*∀*y*(R*xy* → F*x*)
   ∀*x*(F*x* → G*x*)
   ∴∃*x*∃*y*~R*xy*
10. ∀*x*((F*x* v G*x*) → H*x*)
    ∀*x*((H*x* v D*x*) → ~F*x*)
    ∴∀*x*~F*x*

B. Prove that the following pairs of sentences are logically equivalent as was done on pages 290 to 293 in this chapter.

1.	∀*x*(F*x* & G*x*)	(∀*x*F*x* & ∀*x*G*x*)
2.	∀*x*(F*x* & A)	(∀*x*F*x* & A)
3.	∃*x*(F*x* v G*x*)	(∃*x*F*x* v ∃*x*G*x*)
4.	∀*x*(F*x* v A)	(∀*x*F*x* v A)
5.	∀*x*(A → F*x*)	(A → ∀*x*F*x*)
6.	∃*x*(F*x* → A)	(∀*x*F*x* → A)
7.	∀*x*∀*y*F*xy*	∀*y*∀*x*F*xy*
8.	∃*x*∃*y*F*xy*	∃*y*∃*x*F*xy*
9.	~∀*x*∃*y*F*xy*	∃*x*∀*y*~F*xy*
10.	∀*x*((F*x* v G*x*) → H*x*)	∀*x*((F*x* → H*x*) & (G*x*→ H*x*))

C. Some sentences are derivable from no premises at all. These sentences are called theorems of logic, and if our rules are correctly chosen, they will be universally valid sentences. Let us prove a theorem of logic.

To prove: $(\exists y\forall xFxy \rightarrow \forall x\exists yFxy)$

	/BEGIN: $\rightarrow$ INTRO to derive conclusion/	
*1.	$\exists y\forall xFxy$	:ASSUMPTION
	/BEGIN: $\exists$ELIM/	
**2.	$\forall xFxa$	:ASSUMPTION for $\exists$ELIM,1
**3.	Fba	:$\forall$ELIM,2
**4.	$\exists yFby$	:$\exists$INTRO,3
	/END: $\exists$ELIM/	
*5.	$\exists yFby$	:RETURN,4
*6.	$\forall x\exists yFxy$	:$\forall$INTRO,5
*7.	$(\exists y\forall xFxy \rightarrow \forall x\exists yFxy)$	:$\rightarrow$ INTRO,1,6
	/END: $\rightarrow$ INTRO/	
8.	$(\exists y\forall xFxy \rightarrow \forall x\exists yFxy)$	:RETURN,7

Notice that in the proof of a theorem of logic, the last line is not starred. Prove the following theorems of logic:

1. $\exists x(Fx \rightarrow \forall xFx)$
2. $((\exists xFx \rightarrow \forall xFx) \rightarrow (\forall xFx \vee \forall x{\sim}Fx))$
3. $(\forall x(Fx \rightarrow Gx) \rightarrow (\exists x{\sim}Gx \rightarrow \exists x{\sim}Fx))$
4. $(\exists x(\exists yFy \rightarrow Gx(\exists xFx \rightarrow \exists yGy))$
5. ${\sim}\exists y\forall x(Fxy \leftrightarrow {\sim}Fxx)$

D. Symbolize and then construct derivations for these arguments.

1. All phenomenalists deny the reality of matter, but no materialist does. Hence, no materialist is a phenomenalist.
2. No capitalists are socialists. Only socialists are egalitarians. Therefore, no capitalist is an egalitarian.
3. All politicians are good communicators. Some women are politicians. Thus, some women are good communicators.
4. All students take either logic or mathematics. Some students do not take mathematics. Therefore, some students take logic.
5. Anyone who helps a criminal is guilty. Therefore, any criminal who helps himself is guilty.
6. If Adam graduates, then everyone does. Adam graduates only if Betty does also. But Betty graduates only if everyone does. So, if someone doesn't graduate, neither Adam nor Betty graduates.
7. No one who thinks for himself or herself supports every position of the party. One is totally loyal only if one supports every position of the party. Hence, those who are totally loyal do not think for themselves.
8. Some teachers are admired by all those students who admire any teacher at all. Every student admires some teacher or other. Therefore, there are teachers who are admired by all students.

9. People like anything liked by anyone they like. Not everybody dislikes everybody. People like those who like them. Consequently, somebody likes himself.
10. Whenever there is a problem at the college, all the faculty blame the dean for it. Now, if someone blames someone for something, then he (or she) must think that person has control over what he (or she) is being blamed for. The dean is a person. Hence, there is a person whom the faculty thinks has control over all the problems at the college.