Chapter Four
Many-Place Predicates

In chapter 3 we studied the concept of formal validity in the Monadic Predicate Calculus, validity that is due to the logical forms that can be expressed using names, variables, connectives, quantifiers, and one-place predicates. In this chapter the restriction to monadic (one-place) predicates is lifted. We are now focusing simply on Predicate Calculus formal validity: validity that is due to the logical forms that can be expressed using logical signs plus predicates of any number of places.

1 MANY-PLACE PREDICATES

In earlier chapters we used predicate letters that combine with one name to make a sentence:

Antarctica is peaceful	Ea
Fido is a giraffe	Gf
Cynthia ran	*A*c

There are also expressions that combine with two names to form sentences:

Andria is taller than Bill
Cynthia is a friend of David
Fred sees Bella

or with three names, or more:

Cary gave Fido to Andy
Egbert sent Beatrice to Compton
Fred drove Anna to Chicago with David

To accommodate these expressions we use predicate letters that are followed by two or more names or variables enclosed in parentheses. Some examples with names are:

Andria is taller than Bill	T(ab)
Cynthia is a friend of David	F(cd)
Fred sees Bella	S(fb)
Cary gave Fido to Andy	G(cfa)
Egbert sent Beatrice to Compton	S(ebc)

These are atomic sentences, on a par with atomic sentences consisting of a single sentence letter or of a predicate letter followed by a name. They also occur with variables to form atomic formulas:

Andria is taller than x	T(ax)
x *is a friend of* y	F(xy)
z sees Bella	S(zb)
Cary gave x *to* y	G(cxy)
z *sent* u *to* v	S(zuv)

We can use any capital letter for a many-place predicate (adding subscripts if desired). You can tell whether a predicate letter is being used as a one-place predicate or a many-place predicate by seeing what follows it. If it is followed by a single name or variable, it is being used as a one-place predicate; if it is followed by a pair of parentheses containing names or variables, it is being used as a many-place predicate.

An atomic formula is now either a sentence letter alone, or a one-place predicate letter followed by a name or variable, or any predicate letter followed by a pair of parentheses containing any number of names or variables.

These new atomic formulas combine with connectives and quantifiers as in the previous chapter, yielding formulas such as:

∃xA(bx)
∀yB(yy)
∀x(Ax → B(xd))
etc

We are now using parentheses for two different purposes: to surround the terms following a many-place predicate symbol, and to surround molecular formulas. It is common to use either parentheses or square brackets for the latter purpose, and using square brackets instead of parentheses sometimes increases readability. So we will often write complex formulas as follows:

∀x[Ax → B(xd)]
∀x∃y[[A(xy)→B(yx)] ↔ A(yx)∧B(yx)]
∃xF(ax) ↔ ∃y[G(ay) ∧ G(yx)]

Our official account of formulas is now:

A sentence letter is any capital letter between 'P' and 'Z' (perhaps with a subscript).

A one-place predicate is any capital letter between 'A' and 'O' (perhaps with a subscript).

A many-place predicate is any capital letter between 'A' and 'Z' (perhaps with a subscript).

An atomic formula is:

a sentence letter alone,
a one-place predicate letter followed by one name or one variable, or
any predicate letter followed by a pair of parentheses containing any number of names or variables.

If '□' and '△' are formulas, so are:

~□
(□∧△)
(□∨△)
(□→△)
(□↔△)
∃x□
∀x□

EXERCISES

1. Which of the following are formulas in official notation? Which are formulas in informal notation? Which are not formulas at all?

a. ~~F(xa)
b. [∀xG(bx) → ~∃yG(yx)]
c. ∀xG(bx) → ~∃yG(yx)
d. ~Fa ∧ ~G(aa) ∧~Fb ∧ Gxb
e. ~F(a) ∨ ~G(ab)
f. ~Fa ∨ ~Gab
g. ~∃x[~Fx → ∀yG(yy)]
h. ∃x∀y~Fxy
i. ∃x∃yF[xy]

2 SYMBOLIZING SENTENCES USING MANY-PLACE PREDICATES

It should be clear how to symbolize atomic formulas, given the practice from chapter 3 and the explanations above. The only additional complexity has to do with the order of the terms. When symbolizing a transitive verb, such as 'likes' it is convenient to use the same order in the symbolization as in the English, so that 'Ann likes Bill' becomes 'L(ab)'. But when there are more places we need to be careful about the order. For example both of these English sentences say the same thing, but they use a different order:

Ann sent Bill her dog   Ann sent her dog to Bill

It is convenient to be explicit about the ordering, by specifying e.g.

S(①②③): ① sent ② to ③

With this understanding, both of the English sentences above would be symbolized:

S(adb)

Symbolizing complex sentences with many-place predicates mostly involves the same techniques as those we used earlier. For example, when there is one quantificational expression and a name, the symbolizations follow the same patterns as before. Some examples:

Every giraffe is happy	$\forall x[Gx \rightarrow Hx]$
Every giraffe sees Fido	$\forall x[Gx \rightarrow S(xf)]$
Some dog is spotted	$\exists x[Dx \wedge Sx]$
Some dog loves Bobby	$\exists x[Dx \wedge L(xb)]$

The pattern is similar when the name is the subject of the sentence:

Fido sees every dog	$\forall x[Dx \rightarrow S(fx)]$
Bobby loves some dog	$\exists x[Dx \wedge L(bx)]$

When there are two quantificational expressions, the translations may often be produced in stages:

Some dog likes every cat   Partial translation: $\exists x[Dx \wedge$ x likes every cat$]$

Then 'x likes every cat' is handled just as if 'x' were a name:

x *likes every cat* $\forall y[Cy \rightarrow L(xy)]$

The whole sentence then has the form:

$\exists x[Dx \wedge$ x likes every cat $]$

Some dog likes every cat $\exists x[Dx \wedge \forall y[Cy \rightarrow L(xy)]]$

Some additional examples are:

Some dog chased a cat	$\exists x[Dx \wedge$ x *chased a cat*$]$	$\exists x[Dx \wedge \exists y[Cy \wedge H(xy)]]$
Some dog chased no cat	$\exists x[Dx \wedge$ x *chased no cat*$]$	$\exists x[Dx \wedge {\sim}\exists y[Cy \wedge H(xy)]]$
Every dog chased a cat	$\forall x[Dx \rightarrow$ x *chased a cat*$]$	$\forall x[Dx \rightarrow \exists y[Cy \wedge H(xy)]]$
Every dog chased no cat	$\forall x[Dx \rightarrow$ x *chased no cat*$]$	$\forall x[Dx \rightarrow {\sim}\exists y[Cy \wedge H(xy)]]$

Some examples with three-place predicates, using 'G(①②③)' for '① gave ② to ③':

Some nurse gave a doll to a child
$\exists x[Nx \wedge$ x *gave a doll to a child*$]$
$\exists x[Nx \wedge \exists y[Dy \wedge$ x gave y to a child$]]$
$\exists x[Nx \wedge \exists y[Dy \wedge \exists z[Cz \wedge$ x gave y to z$]]]$
$\exists x[Nx \wedge \exists y[Dy \wedge \exists z[Cz \wedge G(xyz)]]]$

No child gave a doll to a nurse
~∃x[Cx ∧ x *gave a doll to a nurse*]
~∃x[Cx ∧ ∃y[Dy ∧ x *gave* y *to a nurse*]]
~∃x[Cx ∧ ∃y[Dy ∧ ∃z[Nz ∧ x *gave* y *to* z]]]
~∃x[Cx ∧ ∃y[Dy ∧ ∃z[Nz ∧ G(xyz)]]]

Every child gave some doll to every nurse
∀x[Cx → x *gave some doll to every nurse*]
∀x[Cx → ∃y[Dy ∧ x *gave* y *to every nurse*]]
∀x[Cx → ∃y[Dy ∧ ∀y[Ny → x *gave* y *to* z]]]
∀x[Cx → ∃y[Dy ∧ ∀y[Ny → G(xyz)]]]

Sometimes in English the wording is unclear regarding which quantificational expression has wider scope; that is, which quantificational expression has the other within its scope. For example, the sentence:

Some freshman dated every sophomore

can be read in two ways. One is the "super-dater" reading, which says that there is a certain freshman who dated every sophomore. Its symbolization is:

∃x[Fx ∧ x *dated every sophomore*]   ∃x[Fx ∧ ∀y[Oy → D(xy)]]

Here, the quantifier '∃x' which originates from the '*some freshman*' has widest scope, and the '∀y' which originates from the '*every sophomore*' is within the scope of '∃x'.

The other reading expresses the more natural situation, which merely says that for every sophomore, some freshman dated him/her:

∀y[Oy → *some freshman dated* y]   ∀y[Oy → ∃x[Fx ∧ D(xy)]]

In this symbolization the '∀y' which originates from the '*every sophomore*' has widest scope, and the quantifier '∃x' which originates from the '*some freshman*' is within the scope of '∀y'.

A slightly more complicated example of this is:

Every ambulance went to a location in a mall

Using ‘I(①③)’ for ‘① is in ②’ and ‘W(①②)’for ‘① went to ②’, this could mean that all the ambulances went to the same location:

∃y[My ∧ ∃x[Lx ∧ I(xy) ∧ ∀z[Az → W(zx)]]]
"*there is a mall, and a location in it, and every ambulance went there*"

This gives the '∃y' widest scope, and within its scope the '∃x' has wider scope than the '∀z'. Or it could mean that they were sent to locations in the same mall, though not necessarily to the same location:

∃y[My ∧ ∀z[Az → ∃x[Lx ∧ I(xy) ∧ W(zx)]]]
"*there is a mall and every ambulance was sent to some location in it*"

In this symbolization the '∃y' still has widest scope, but the '∃x' and '∀z' are interchanged. Or it could merely mean that each ambulance was sent to some location in some mall:

∀z[Az → ∃y[My ∧ ∃x[Lx ∧ I(xy) ∧ W(zx)]]]
"*every ambulance is such that there is a mall and a location in it and the ambulance went there*"

In this symbolization the '∀z' now has widest scope, and the '∃x' is within the scope of the '∃y'. All three symbolizations have the same ingredients; they differ with respect to how those ingredients are arranged.

Certain words have as their main function indicating that the quantifier they occur with has a wide scope. An example is 'certain' in:

Every reporter admired a certain car.

The '*certain*' gives the existential quantifier with 'car' wide scope:

$\exists x[Cx \wedge \forall y[Ey \rightarrow A(yx)]]$

The phrase '*the same*' can also work in this way, as in:

Every reporter admired the same car.

$\exists x[Cx \wedge \forall y[Ey \rightarrow A(yx)]]$

At every hoe-down the same fiddler played the same song
∃x[*fiddler* x ∧ ∃y[*song* y ∧ *at every hoe-down* x *played* y]]
∃x[*fiddler* x ∧ ∃y[*song* y ∧ ∀z[hoe-down z → x *played* y *at* z]]]
$\exists x[Fx \wedge \exists y[Gy \wedge \forall z[Hz \rightarrow P(xyz)]]]$

As in earlier chapters, some linguistic constructions have no obvious rationale in terms of their parts. An example is the medieval example '*No man lectures in Paris unless he is a fool*'. Here are different but equivalent symbolizations (using 'L(①②)' for '① *lectures in* ②' and 'a' for '*Paris*'):

$\forall x[Mx \rightarrow \sim L(xa) \vee Fx]$
$\sim \exists x[Mx \wedge L(xa) \wedge \sim Fx]$

EXERCISES

1. Symbolize each of the following:

a. *Hans sees every doctor but Amanda doesn't see any doctor.*
b. *Hans, who owns a dog, doesn't own a cat.*
c. *Hans loves Amanda but she doesn't love him.*
d. *Neither Hans nor Amanda has a cat.*
f. *Some hyena and some giraffe like each other*.
g. *Some giraffe likes every baboon.*
h. *Some giraffe that likes every baboon likes no hyena.*
i. *Some giraffe likes every baboon that likes no hyena*
j. Some giraffe likes every baboon that likes it
k. *Eileen resides in a big city*. <use 'R(①②)' for '① resides in ②'>
l. *Eileen and Betty both reside in the same city*.
m. *If Hank resides in Brea then he attends UCLA; otherwise he doesn't attend UCLA.*
n. *If David and Hank both live in Brea then David attends a private school and Hank attends a public school.*
o. *Nobody who comes from Germany attends a California school.*
p. *No giraffe likes Fido unless it is crazy*
q. *Nobody gives a book to a freshman unless it is inexpensive*

3 DERIVATIONS

Adding many-place predicates to the notation has no effect on the rules of inference; they are already adequate as they stand. Here are two examples, using familiar techniques.

Any giraffe that is taller than Harriet is taller than every zebra. Some giraffes aren't taller than some zebras. So there is a giraffe that is not taller than Harriet.

∀x[Gx ∧ T(xh) → ∀y[Ey → T(xy)]]
∃x[Gx ∧ ∃y[Ey ∧ ~T(xy)]]
∴ ∃x[Gx ∧ ~T(xh)]

1. ~~Show~~ ∃x[Gx ∧ ~T(xh)]

2.	Gu ∧ ∃y[Ey ∧ ~T(uy)]	pr2 ei
3.	Gu	2 s
4.	∃y[Ey ∧ ~T(uy)]	2 s
5.	Ev ∧ ~T(uv)	4 ei
6.	Gu ∧ T(uh) → ∀y[Ey → T(uy)]	pr1 ui
7.	~~Show~~ ~T(uh)	
8.	T(uh)	ass id
9.	Gu ∧ T(uh)	3 8 adj
10.	∀y[Ey → T(uy)]	6 9 mp
11.	Ev → T(uv)	10 ui
12.	T(uv)	5 s 11 mp
13.	~T(uv)	5 s 12 id
14.	Gu ∧ ~T(uh)	3 7 adj
15.	∃x[Gx ∧ ~T(xh)]	14 eg dd

Betty scolded every dog that chased a cat. Betty is a jeweler. Some dog that chased Cleo was grey. Cleo is a cat. So a jeweler scolded some grey dog.

∀x[Dx ∧ ∃y[Cy ∧ H(xy)] → S(bx)]
Jb
∃x[Dx ∧ H(xc) ∧ Gx]
Cc
∴ ∃x[Jx ∧ ∃y[Dy ∧ Gy ∧ S(xy)]]

1. ~~Show~~ ∃x[Jx ∧ ∃y[Dy ∧ Gy ∧ S(xy)]]

2.	Du ∧ H(uc) ∧ Gu	pr3 ei
3.	Du ∧ ∃y[Cy ∧ H(uy)] → S(bu)	pr1 ui
4.	H(uc)	2 s s
5.	Cc ∧ H(uc)	pr4 4 adj
6.	∃y[Cy ∧ H(uy)]	5 eg
7.	Du	2 s s
8.	Du ∧ ∃y[Cy ∧ H(uy)]	7 6 adj
9.	S(bu)	8 3 mp
10.	Du ∧ Gu	2 s 7 adj
11.	Du ∧ Gu ∧ S(bu)	9 10 adj
12.	∃y[Dy ∧ Gy ∧ S(by)]	11 eg
13.	Jb ∧ ∃y[Dy ∧ Gy ∧ S(by)]	pr2 12 adj
14.	∃x[Jx ∧ ∃y[Dy ∧ Gy ∧ S(xy)]	13 eg dd

When existential quantifiers combine with universal ones, in general, a formula beginning with an existential followed by a universal is stronger than a universal followed by an existential. This simple derivation illustrates this:

	∃x∀yF(xy)	something forces everything
∴	∀y∃xF(xy)	everything is forced by something

1. ~~Show~~ ∀y∃xF(xy)
2. ~~Show~~ ∃xF(xy)
3. ∀yF(uy) — pr1 ei
4. F(uy) — 3 ui
5. ∃xF(xy) — 4 eg dd
6. — 2 ud

If we try to prove the reverse:

∀y∃xF(xy)
∴ ∃x∀yF(xy)

we won't be able to. The obvious strategy would be to set up a universal derivation, trying to derive 'F(uy)' in order to show '∀yF(uy)'. If we could do this, we could existentially generalize, and the derivation would be done. So we set things up for that:

1. Show ∃x∀yF(xy)
2. Show ∀yF(xy)
3. ∃xF(xy) — pr1 ui
4. F(uy) — 3 ei ud?????

But this is not in the right form for a universal derivation; there is an 'F(uy)' when a 'F(xy)' is needed. And there is no way to get 'F(xy)' from '∃xF(xy)' by ei, since ei requires that the variable be new. One might naturally then try to derive the conclusion indirectly, by an indirect derivation:

1. Show ∃x∀yF(xy)
2. ~∃x∀yF(xy) — ass id
3. ∀x~∀yF(xy) — 2 qn
4. ~∀yF(xy) — 3 ui
5. ∃y~F(xy) — 4 qn
6. ∃xF(xy) — pr1 ui
7. ??????????

The problem at this point is that when one uses ei on lines 5 and 6, different variables need to be used, and there seems to be no way to derive a contradiction. In fact, the argument is invalid, and so no derivation will work. This will be shown in section 9.

STRATEGY HINTS: The strategy hints given at the end of chapters 2 and 3 remain unchanged. They are repeated here for convenience.

Try to reason out the argument for yourself.

Begin with a sketch of an outline of a derivation, and then fill in the details.

Write down obvious consequences.

When no other strategy is obvious, try indirect derivation.

To derive:	Try this:
Conjunction **□ ∧ ○**	Derive each conjunct, and adjoin them
Disjunction **□ ∨ ○**	Derive either disjunct and use add. (Often this is not possible.) Assume '~(□ ∨ ○)' for id and immediately use dm. Derive '~□ → ○' and use cdj
Conditional **□ → ○**	Use cd
Biconditional **□ ↔ ○**	Derive both conditionals and use cb.
Negation of conjunction **~(□ ∧ ○)**	Use id.
Negation of disjunction **~(□ ∨ ○)**	Derive '~□ ∧ ~○' and use dm. Perhaps assume '□ ∨ ○' for id and try to derive both '□ → P∧~P' and '○→P∧~P'. Then use sc (applied to the assumed '□ ∨ ○' and the conditionals) to derive 'P∧~P'.
Negation of conditional **~(□ → ○)**	Use id.
Negation of biconditional **~(□ ↔ ○)**	Derive '□ ↔ ~○' and use nb.

If you have this available:	Try this:
Conjunction **□ ∧ ○**	Simplify and use the conjuncts singly.
Disjunction **□ ∨ ○**	Try to derive the negation of one of the disjuncts, and use mtp.. Derive the conditionals '□ → △' and '○ → △', where '△' is something you want to derive. Then use sc with the disjunction and two conditionals.
Conditional **□ → ○**	Try to derive the antecedent to set up mp, or derive the negation of the consequent, to set up mt.
Biconditional **□ ↔ ○**	Infer both conditionals and use them with mp, mt, and so on.
Negation of conjunction **~(□ ∧ ○)**	Use dm to turn this into '~□ ∨ ~○', and then try to derive either '□' or '○' to use mtp.
Negation of disjunction **~(□ ∨ ○)**	Use dm to turn this into '~□ ∧ ~○'; then simplify and use the conjuncts singly.

Negation of conditional **~(□ → ○)**	Use nc to derive '□ ∧ ~○', then simplify and use the conjuncts singly.
Negation of biconditional **~(□ ↔ ○)**	Use nb to turn this into '□ ↔ ~○', and use bc to get the corresponding conditionals.

To derive:	**Try this:**
Universal Quantification **∀x□**	Set up a universal derivation. Write a show line containing ∀x□, and then immediately follow this with a show line containing □. When the second show is cancelled, use rule ud to cancel the first. Or write a show line with '∀x□', and then assume '~∀x□' for an indirect derivation. Turn this into '∃x~□', and proceed from there.
Existential Quantification **∃x□**	Derive an instance and then use rule eg. Or write a show line with '∃x□', and then assume '~∃x□' for an indirect derivation. Turn this into '∀x~□', and proceed from there.
Negation of a Universal Quantification **~∀x□**	State a show line with '~∀x□', and then assume '∀x□' for an indirect derivation. Or derive '∃x~□' and apply derived rule qn.
Negation of an Existential Quantification **~∃x□**	State a show line with '~∃x□', and then assume '∃x□' for an indirect derivation. Or derive '∀x~□' and apply derived rule qn.

If you have this available:	**Try this:**
Universal Quantification **∀x□**	Use rule ui to derive an instance. (But use rule ei first if that is an option.)
Existential Quantification **∃x□**	Use rule ei to derive an instance.
Negation of a Universal Quantification **~∀x□**	Use derived rule qn to turn this into an existential quantification.
Negation of an Existential Quantification **~∃x□**	Use derived rule qn to turn this into a universal quantification.

Use rule av if necessary: If you are having difficulty with capturing when you use rule ui or ei, change what you are trying to derive to an alphabetic variant. Complete the derivation, and then use derived rule av to convert this into a derivation of what you are after.

EXERCISES

Show each of the following arguments to be valid.

1. $\forall x \forall y \forall z[S(xy) \wedge S(yz) \rightarrow S(xz)]$
   $S(bc) \wedge S(ab)$
   $\therefore$ $S(ac)$

2. $\forall x \forall y[Ax \wedge By \rightarrow [S(xy) \leftrightarrow \sim S(yx)]]$
   $\therefore$ $\forall x \forall y[Ax \wedge By \rightarrow [S(xy) \vee S(yx)]]$

3. $\forall x \exists y S(xy)$
   $\forall x \forall y[Cx \wedge S(xy) \rightarrow Dy]$
   $\forall x \forall y[Dx \wedge S(yx) \rightarrow Dy]$
   $\therefore$ $\sim \exists x[Cx \wedge \sim Dx]$

4. $\exists x Ex \wedge \exists x \sim Ex$
   $\forall x \forall y[Ex \wedge S(xy) \rightarrow Ey]$
   $\therefore$ $\exists x \exists y \sim S(xy)$

5. $\forall x \forall y[S(xy) \leftrightarrow S(yx)]$
   $\exists x \exists y[Ax \wedge By \wedge S(xy)]$
   $\therefore$ $\exists x \exists y[By \wedge Ax \wedge S(yx)]]$

6. $\exists x[Ax \wedge \forall y[By \rightarrow S(xy)]]$
   $\forall x \forall y[Bx \leftrightarrow Cy]$
   $\therefore$ $\forall x[Cx \rightarrow \exists y S(yx)]$

7. Prove the following biconditional theorems.

T251 $\boldsymbol{\forall x \forall y F(xy) \leftrightarrow \forall y \forall x F(xy)}$ <universal quantifiers permute>

T252 $\boldsymbol{\exists x \exists y F(xy) \leftrightarrow \exists y \exists x F(xy)}$ <existential quantifiers permute>

T255 $\boldsymbol{\forall x \forall y F(xy) \leftrightarrow \forall y \forall x F(yx)}$ <multiple applications of av>

T261 $\boldsymbol{\forall x[\exists y F(xy) \rightarrow \exists y G(xy)] \leftrightarrow \forall x \forall y \exists z[F(xy) \rightarrow G(xz)]}$

4 THE RULE "INTERCHANGE OF EQUIVALENTS"

Although no new rules are needed when many-place predicates are added, some new rules are convenient. One of these is called "Interchange of Equivalents". This rule allows us to change any part of a formula to a known equivalent part. For example, it allows us to change:

∀x[Gx ∨ ~~H(xx)]

into:

∀x[Gx ∨ H(xx)]

by changing the part '~~H(xx)' to the equivalent 'H(xx)'. We know that '~~H(xx)' is equivalent to 'H(xx)' because rule dn says so. This new rule is given by:

> **Rule ie** ("interchange of equivalents") <preliminary statement>
>
> If we have a rule stating that a certain formula '□' is equivalent to another formula '○', then from any available line whose formula contains '□' we may infer a line with the same formula but with '□' changed to '○'. The justification consists of writing 'ie' followed by '/' and the name of the rule giving the equivalence.
>
> A rule establishes that one formula is equivalent to another if the rule can be applied to either to infer the other. These rules all establish equivalents:
>
> dn <double negation>
> nc <negation of conditional>
> cdj <conditional/disjunction>
> dm <demorgan's>
> nb <negation of biconditional>
> qn <quantifier negation>
> av <alphabetic variation>
>
> Plus any rules based on a theorem that is biconditional in form.

Here is an example of a derivation using the new rule:

∀x[~ [Ax ∨ Bx] ↔ ∃y~Hy]
∴ ∀x[~Ax ∧ ~Bx ↔ ~∀yHy]

1. ~~Show~~ ∀x[~Ax ∧ ~Bx ↔ ~∀yHy]

2.	∀x[~ [Ax ∨ Bx] ↔ ~∀yHy]	pr1 ie/qn
3.	∀x[~Ax ∧ ~Bx ↔ ~∀yHy]	2 ie/dm dd

Step 2 uses one of the cases of quantifier negation to change '∃y~Hy' in the premise to '~∀yHy'. Step 3 then uses a case of De Morgan's law to change '~ [Ax ∨ Bx]' on line 2 to '~Ax ∧ ~Bx'.

This rule also works when the equivalence of the formulas being replaced is given to us by a premise or an earlier available line, as in this derivation:

P ↔ Q ∧ R
S ↔ ~P
∴ S → ~[Q∧R]

1. ~~Show~~ S → ~[Q∧R]

2.	S ↔ ~[Q∧R]	pr2 ie/pr1
3.	S → ~[Q∧R]	2 bc dd

Here the first premise tells us that 'P' and 'Q∧R' are equivalent, so on line 2 we may replace 'P' in the second premise by 'Q∧R'. (Step 3 is already familiar.) A full statement of rule IE is:

Rule ie ("interchange of equivalents")

If we have a rule stating that a certain formula '□' is equivalent to another formula '○', then from any available line whose formula contains '□' we may infer a new line with the same formula but with '□' changed to '○'. The justification consists of writing 'ie' followed by '/' and the name of the rule giving the equivalence.

A rule establishes that one formula is equivalent to another if the rule can be applied to either to infer the other. These rules all establish equivalents:

dn	<double negation>
nc	<negation of conditional>
cdj	<conditional/disjunction>
dm	<demorgan's>
nb	<negation of biconditional>
qn	<quantifier negation>
av	<alphabetic variation>

Plus any theorem that is biconditional in form.

Also:

If we have a premise or available line that is a biconditional of the form '□ ↔ ○', or '○ ↔ □', then from any available line whose formula contains '□' we may infer a line with the same formula but with '□' changed to '○'. The justification consists of writing 'ie' followed by '/' and the name of the premise or line used.

EXERCISES

In the following derivations, several lines appeal to the rule for interchanging equivalents. Say which of these lines are correct, and which incorrect. (In each case, when judging a given line assume that all previous lines are OK.)

1. $P \leftrightarrow Q \lor R$
   $\sim Q \rightarrow \sim S \lor P$
   $\therefore$ $R \lor \sim Q$

1. ~~Show~~ $R \lor \sim Q$

2.	$\sim\sim P \leftrightarrow Q \lor R$	pr1 ie/dn
3.	$\sim\sim P \leftrightarrow P$	2 ie/pr2
4.	$\sim\sim P$	3 ie/dn
5.	P	4 ie/dn
6.	$\sim S \lor P$	5 add
7.	$\sim Q$	6 ie/pr2
8.	$R \lor \sim Q$	7 add dd

2. ∀x∃y[Ax ↔ R(xy)]
∀z∀y[R(zy) ↔ S(yz)]
∀x[[Ax↔Ax] ↔ Ax]
∴ Au

1. ~~Show~~ Au

2.	∃y[Ax ↔ R(xy)]	pr1 ui
3.	Au ↔ R(xu)	2 ei
4.	R(xu) ↔ S(ux)	pr2 ui ui
5.	Au ↔ S(ux)	4 ie/3
6.	Au ↔ S(ux)	3 ie/4
7.	Au ↔ Au	5 ie/6
8.	[Au↔Au] ↔ Au	pr3 ui
9.	Au	7 ie/8 dd

A strengthened form of rule ie is also available. Suppose that you are given the following as a theorem or as a premise or a formula on an available line:

∀x∀y∀z(□↔○)

Then you may use rule ie to replace '□' by '○' within a formula on an available line even if the varialbles 'x', 'y', and 'z' are bound in that formula. For example if you are in a derivation with the following pattern:

7. ∀x∀y(R(xy) ↔ Gx ∧ ~Hy)
:::::
:::::
13. ∀x∃y(~R(xy) ∧ Py)

Then you may replace 'R(xy)' in line 13 to get:

14. ∀x∃y(~(Gx ∧ ~Hy) ∧ Py) 13 ie/7

(In this explanation we have chosen three particular variables for illustration; any number may be used.)

A constraint on this rule is that there must be no other variables free in '□' or '○' which become bound when the substitution is made.

This rule applies even when the variables used on the later line are different from those in the biconditional line, so long as the biconditional line could be changed into a biconditional with the same variaibles by repeated use of rule av in conjunction with ie. So given lne 7 as above, this would also be an allowable move:

13. ∀u∃v(~R(uv) ∧ Pv)

14. ∀u∃v(~(Gu ∧ ~Hv) ∧ Pv) 13 ie/7

5 BICONDITIONAL DERIVATIONS

When proving a biconditional informally, people sometimes give a string of equivalences. For example, to show informally that this is a theorem:

∴ ~P ∧ ~~Q ↔ ~[P ∨ ~Q]

one might reason as follow:

By double negation, '~P ∧ ~~Q ' is equivalent to '~P ∧ Q',
and by De Morgan's laws, that is equivalent to '~[~~P ∨ ~Q]',
and again by double negation, that is equivalent to '~[P ∨ ~Q]'.
So the first (namely '~P ∧ ~~Q') is equivalent to the last (namely, '~[P ∨ ~Q]').

You can establish a biconditional by showing that one of its sides is equivalent to something, which is equivalent to some further thing, etc, ending up with the other side of the biconditional. This idea can be implemented by means of a new technique, called "biconditional derivation". It goes as follows:

Biconditional Derivations

Any show line with a biconditional formula '□ ↔ ○' may be followed by an assumption consisting of a line containing either '□', or '○', justified by the notation 'ass bd' (meaning, "assumption for a biconditional derivation").

A derivation may be continued so that each additional step follows from the immediately preceding step by rule IE, so that eventually you reach a line containing '○' or '□' (whichever was not on the assumption line). Then 'bd' may be written at the end of the last line; box all lines starting with the assumption line, and cancel the 'show'.

(Alternative: As usual, you may end the derivation by writing an empty line following the line containing '○' or '□', writing the line number of the previous line and 'bd'; then you box and cancel.)

Here is a derivation for the example above:

∴ ~P ∧ ~~Q ↔ ~[P ∨ ~Q]

1. Show ~P ∧ ~~Q ↔ ~[P ∨ ~Q]

2.	~P ∧ ~~Q	ass bd
3.	~P ∧ Q	2 ie/dn
4.	~[~~P ∨ ~Q]	3 ie/dm
5.	~[P ∨ ~Q]	4 ie/dn bd

Line 1 contains the biconditional to be shown. Line 2 assumes its left-hand side for the purpose of a biconditional derivation. Lines 3-5 make inferences of formulas equivalent to that on line 2. Since this series of IE steps ends up with the right-hand side of the biconditional on the show line, we conclude the derivation, boxing and canceling.

The alternative form of the derivation is to delay until line 6 the "bd" justification with boxing and cancelling:

1. Show ~P ∧ ~~Q ↔ ~[P ∨ ~Q]

2.	~P ∧ ~~Q	ass bd
3.	~P ∧ Q	2 ie/dn
4.	~[~~P ∨ ~Q]	3 ie/dm
5.	~[P ∨ ~Q]	4 ie/dn
6.		5 bd

This new derivation technique is not essential, since whatever we can do with it we can also do without it. But doing without it requires a longer derivation -- sometimes a much longer derivation. In particular, we can always derive the biconditional by giving two conditional derivations, followed by an application of rule cb. Following this pattern, we can convert the above derivation to one without rule bd as follows:

1.	Show ~P ∧ ~~Q ↔ ~[P ∨ ~Q]	
2.	Show ~P ∧ ~~Q → ~[P ∨ ~Q]	
3.	~P ∧ ~~Q	ass cd
4.	~P ∧ Q	3 ie/dn
5.	~[~~P ∨ ~Q]	4 ie/dm
6.	~[P ∨ ~Q]	5 ie/dn cd
7.	Show ~[P ∨ ~Q] → ~P ∧ ~~Q	
8.	~[P ∨ ~Q]	ass cd
9.	~[~~P ∨ ~Q]	8 ie/dn
10.	~P ∧ Q	9 ie/dm
11.	~P ∧ ~~Q	10 ie/dn cd
12.	~P ∧ ~~Q ↔ ~[P ∨ ~Q]	2 7 cb dd

Clearly, using biconditional derivation simplifies matters considerably. (The derivation could also be done without any use of rule ie as well; this would make it much longer.)

It is important when applying rule bd that every step after the assumption is justified by rule ie. If other rules are used, then even though every line follows correctly by an established rule, one cannot apply rule bd to box and cancel. This is because bd derives an equivalence, and so only lines that infer equivalences of previous lines are permitted. For example, the last line of this derivation is incorrect:

∴ ~P ∧ ~~Q ↔ ~P <*Invalid argument*>

1.	Show ~P ∧ ~~Q ↔ ~P		
2.	~P ∧ ~~Q	ass bd	
3.	~P ∧ Q	2 ie/dn	
4.	~P	3 s	
5.		4 bd	← ***incorrect***

Line 5 is incorrect because a rule other than ie is used to get line 4 from line 3. The derivation is thus incorrect -- which is good, since the sentence that it purports to derive from no premises is not a tautology; it is false when 'P' and 'Q' are both false.

EXERCISES

1. Prove the biconditional above without using a biconditional derivation and also without using the rule for interchange of equivalents:

∴ ~P ∧ ~~Q ↔ ~[P ∨ ~Q]

2. Prove the following, using a biconditional derivation.

T250 ∀x∀y∀zF(xyz) ↔ ~∃x∃y∃z~F(xyz) <a generalization of qn>

6 SENTENCES WITHOUT OVERLAY OF QUANTIFIERS

When showing invalidities in chapter 3 we saw that it is sometimes difficult to assess the truth values of formulas, particularly when their quantifiers have overlapping scopes, although when the quantifiers do not have overlapping scopes it is easy. For example, given that Agatha is happy but not carefree, and that Beatrice is carefree but not happy, and that they are the only two things in the universe, it is easy to evaluate certain kinds of quantified formulas, such as:

∃xHx	true, because Agatha is happy
∃x~Hx	true, because Beatrice isn't happy
∃xCx ↔ ∀yHy	false because '∃xCx' is true, and '∀yHy' is false

but this is harder to assess:

∀x∃y[Cx ↔ Hy]	This sentence is true. It's true because for anyone you choose, either they're carefree, and there's someone who is happy, and the biconditional is true, or they aren't carefree, and there's someone who isn't happy, and again the biconditional is true

The point is that when one quantifier falls inside the scope of another, especially if one quantifier is universal and the other existential, it can be a sophisticated matter to decide whether the sentence is true or not. This is why truth-functional expansions of formulas, although complex and artificial to produce, can sometimes be helpful in deciding what is true and what is false in a counter-example.

For sentences containing only monadic predicates, there is a way to eliminate the problem cases entirely. This is because when there are no many-place predicates, every formula is provably equivalent to one in which no quantifier falls within the scope of another quantifier. (This is sometimes described as a formula "without overlay", where 'overlay' refers to a situation in which one quantifier contains another within its scope.) The proof of this in any given case can be developed using what in chapter 3 were called *laws of confinement*. Here are some confinement laws, repeated from the previous chapter:

Derived Rule conf <confinement>

'**conf**' refers to any use of any of the following theorems.

T215	∀x[P∧Fx] ↔ P∧∀xFx	or	∀x[Fx∧P] ↔ ∀xFx∧P
T216	∃x[P∧Fx] ↔ P∧∃xFx	or	∃x[Fx∧P] ↔ ∃xFx∧P
T217	∀x[P∨Fx] ↔ P∨∀xFx	or	∀x[Fx∨P] ↔ ∀Fx∨P
T218	∃x[P∨Fx] ↔ P∨∃xFx	or	∃x[Fx∨P] ↔ ∃xFx∨P
T219	∀x[P→Fx] ↔ [P→∀xFx]		
T220	∃x[P→Fx] ↔ [P→∃xFx]		
T221	∀x[Fx→P] ↔ [∃xFx→P]		
T222	∃x[Fx→P] ↔ [∀xFx→P]		

These laws may be applied for any sentence letter in place of 'P', or for any formula that has no free occurrence of 'x' in place of 'P'. They are called "confinement" laws because when 'x' is not free in 'P', a quantifier governing the whole molecular formula may be confined to one part of the formula.

These laws are very useful when used in conjunction with a few other derived rules for commutativity, associativity, distribution, quantifier distribution, and biconditional expansion.

Derived Rule com <commutativity>

'**com**' refers to any use of any of the following theorems.

T24	P∧Q ↔ Q∧P
T53	P∨Q ↔ Q∨P
T92	(P↔Q) ↔ (Q↔P)

Derived Rule assoc <associativity>

'**assoc**' refers to any use of any of the following theorems.

T25	$P\wedge(Q\wedge R) \leftrightarrow (P\wedge Q)\wedge R$
T54	$P\vee(Q\vee R) \leftrightarrow (P\vee Q)\vee R$
T94	$(P\leftrightarrow(Q\leftrightarrow R)) \leftrightarrow ((P\leftrightarrow Q)\leftrightarrow R)$

Derived Rule dist <distribution>

'**dist**' refers to any use of any of the following theorems.

T61	$P\wedge(Q\vee R) \leftrightarrow (P\wedge Q)\vee(P\wedge R)$
T62	$P\vee(Q\wedge R) \leftrightarrow (P\vee Q)\wedge(P\vee R)$
T116	$(P\wedge Q)\vee(R\wedge S) \leftrightarrow (P\vee R)\wedge(P\vee S)\wedge(Q\vee R)\wedge(Q\vee S)$
T117	$(P\vee Q)\wedge(R\vee S) \leftrightarrow (P\wedge R)\vee(P\wedge S)\vee(Q\wedge R)\vee(Q\wedge S)$

Derived Rule qdist <quantifier distribution>

'**qdist**' refers to any use of either of the following theorems.

T207	$\exists x(Fx\vee Gx) \leftrightarrow \exists xFx\vee\exists xGx$
T208	$\forall x(Fx\wedge Gx) \leftrightarrow \forall xFx\wedge\forall xGx$

Derived Rule bex <biconditional expansion>

'**bex**' refers to any use of either of the following theorems.

T81	$(P\leftrightarrow Q) \leftrightarrow (P\to Q)\wedge(Q\to P)$
T83	$(P\leftrightarrow Q) \leftrightarrow (P\wedge Q)\vee(\sim P\wedge\sim Q)$

Derived Rule vac <vacuous quantification>

'**vac**' refers to any use of either of the following theorems.

T227	$\forall xP \leftrightarrow P$
T228	$\exists xP \leftrightarrow P$

Suppose, for example, we are given the sentence '$\forall x\exists y[Fx\wedge Gy]$'. We can prove that this is equivalent to the sentence '$\forall xFx\wedge\exists yGy$'. We do so using a biconditional derivation, using only the confinement laws:

$\therefore\ \forall x\exists y[Fx\wedge Gy] \leftrightarrow \forall xFx\wedge\exists yGy$

1. ~~Show~~ $\forall x\exists y[Fx\wedge Gy] \leftrightarrow \forall xFx\wedge\exists yGy$

2.	$\forall x\exists y[Fx\wedge Gy]$	ass bd
3.	$\forall x[Fx\wedge\exists yGy]$	2 ie/conf
4.	$\forall xFx\wedge\exists yGy$	3 ie/conf bd

Another example: We can eliminate the overlay in:

$\exists x[\forall yFy \to \forall z[Gz\wedge Rx]]$

as follows:

1. ~~Show~~ ∃x[∀yFy → ∀z[Gz ∧ Rx]] ↔ [∀yFy → ∀zGz ∧ ∃xRx]

2.	∃x[∀yFy → ∀z[Gz ∧ Rx]]	ass bd
3.	∀yFy → ∃x∀z[Gz ∧ Rx]	2 ie/conf
4.	∀yFy → ∃x[∀zGz ∧ Rx]	3 ie/conf
5.	∀yFy → ∀zGz ∧ ∃xRx	4 ie/conf bd

Any formula with no many-place predicates can be transformed into a logically equivalent formula in which there is no quantifier overlay. Here is a simple routine for doing so:

First, replace all biconditionals in the formula by combinations of formulas without biconditional signs. For example, convert 'P↔Q' into '[P∧Q]∨[~P∧~Q]' using the ie/bex.

Whenever a quantifier immediately precedes a negation, apply ie/qn to move the quantifier to the right.

When a quantifier immediately precedes a conjunction or disjunction or conditional, you may be able to move the quantifier inside using ie/conf or ie/qdist. For example, using ie/conf:

∃x[P∨Fx]	becomes	P ∨ ∃xFx
∃x[P∧Fx]	becomes	P ∧ ∃xFx
∀x[P∨Fx]	becomes	P ∨ ∀xFx
∀x[P∧Fx]	becomes	P ∧ ∀xFx
∃x[P→Fx]	becomes	P → ∃xFx
∀x[P→Fx]	becomes	P → ∀xFx
∃x[Fx→P]	becomes	∀xFx→P
∀x[Fx→P]	becomes	∃xFx→P

And using ie/qdist:

∃x[Gx∨Fx]	becomes	∃xGx ∨ ∃xFx
∀x[Gx∧Fx]	becomes	∀xGx ∧ ∀xFx

If ie/conf does not apply, and you have a universal quantifier immediately preceding a disjunction, or an existential quantifier immediately preceding a conjunction, you can modify the disjunction or conjunction using ie/dist. If a universal quantifier precedes a disjunction, use ie/dist to turn the disjunction into a conjunction, and ie/qdist then applies; if an existential quantifier precedes a conjunction, use ie/dist to turn the conjunction into a disjunction, and ie/qdist then applies. Examples:

∃x[Fx∧[Gx∨∀yHy]] becomes by ie/dist ∃x[[Fx∧Gx]∨[Fx∧∀yHy]]
and then ie/qdist applies

∀x[Fx∨ [Gx∧∀yHy]] becomes by ie/dist ∀x[[Fx∨Gx]∧[Fx∨∀yHy]]
and then ie/qdist applies

Sometimes it may be necessary to use ie/assoc before applying one of the above rules. For example, suppose you are given '∃x[Fx∧[Gx∧∀yHy]]'. Then none of the above rules apply. But the conjuncts may be reordered:

∃x[Fx∧[Gx∧∀yHy]] becomes by ie/assoc ∃x[[Fx∧Gx]∧∀yHy]
and then ie/conf applies.

Finally, a quantifier may end up having scope over another when it is actually binding nothing at all, as in '∃x∀yFy'. In this case rule ie/vac just lets you drop the quantifier that isn't binding anything:

∃x∀yFy becomes by ie/vac ∀yFy

The good news is that if a formula contains no many-place predicates it is provably equivalent to a formula without overlay, and formulas without overlay are often much easier to assess. The bad news is that when a formula contains at least one many-place predicate, no such equivalence is guaranteed. There are plenty of examples of formulas that are not equivalent to any without overlay. Here are two:

$\forall x \exists y F(xy)$
$\exists y \forall x F(xy)$

Suppose that 'F' stands for loving, and that we are discussing a universe consisting only of people. Then the first says that everyone loves someone, and the second says that there is someone loved by everyone. There is no simpler way to symbolize either of these.

As a general summary of strategy, your goal will be to move quantifiers inside using QN, Qdist, Conf,and Vac, while using Assoc, Com, Dist, DM to reorder the formulas that are within the scopes of the quantifiers so that the former rules will apply.

EXERCISES

1. For each of the following formulas, find an equivalent formula which has no overlay of quantifiers, and prove that it is equivalent. (The derivations are easiest using biconditional derivations.)

a. $\forall z[\exists u[Fu \rightarrow Gz] \rightarrow Fz]$
b. $\exists z \forall x[Fx \leftrightarrow Fz]$
c. $\exists x Fx \vee \forall y{\sim}Gy$
d. $\exists x[\exists x[Fx \leftrightarrow Gx] \rightarrow [Fx \leftrightarrow Gx]]$
e. $\forall x \exists y \forall z[Fx \wedge Gz \rightarrow Fz \vee Gy]$

7 PRENEX FORMS

A formula is in *prenex form* when all of its quantifiers are in a string on the front of the formula, with each quantifier having scope over everything to its right. These formulas are in prenex form:

∀x∃y∃z∀u~[P(xy) ∧ Q(yz)]
∀x∀y∃z[R(xz) → ~S(zy)]
∃x[Hx → Gx∧ Ky]

These are not:

∀x∃y∃z~∃u[P(xy) ∧ Q(yz)]	'∃u' is not part of the string of quantifiers on the front
∀x∀y[R(xz) → ~∃zS(zy)]	'∃z' is not part of the string of quantifiers on the front
∀x∃yR(xy) → Gy	'∀x' and '∃y' do not have scope over the whole formula

Every formula that we can express is logically equivalent to one that is in prenex form. In fact, any formula can easily be converted into prenex form. That is, any formula can be transformed into a logically equivalent formula which is in prenex form. Here is a routine for doing so:

First, replace all biconditionals by combinations of formulas without biconditional signs. For example, convert 'P↔Q' into '[P→Q] ∧ [Q→P]' or '[P∧Q]∨[~P∧~Q]' using the derived rule ie/bex.

Second, use ie/av to change bound variables within the formula so that every quantifier uses a different variable, and so that no quantifier uses the same variable as one that occurs free in the original formula. For example, convert '∀x[Hx ∧ Jy → ∃yK(xy)]' into '∀x[Hx ∧ Jy → ∃wK(xw)]'.

Now move each quantifier to the front of the formula by a series of steps in accordance with these patterns:

If the quantifier is immediately preceded by a negation, move the quantifier to the left of the negation, changing the quantifier from existential to universal, or vice versa. This step is justified by ie/qn.

~∃x	becomes	∀x~
~∀x	becomes	∃x~

The remaining patterns appeal to the confinement laws.

If the quantifier is on the front of a disjunct, move the quantifier to the front of the whole disjunction. This rule is justified by ie/conf.

P ∨ ∃xFx	becomes	∃x[P ∨ Fx]
∃xFx ∨ P	becomes	∃x[Fx ∨ P]
P ∨ ∀xFx	becomes	∀x[P ∨ Fx]
∀xFx ∨ P	becomes	∀x[Fx ∨ P]

If the quantifier is on the front of a conjunct, move the quantifier to the front of the whole conjunction, having scope over it. This rule is justified by ie/conf.

P ∧ ∃xFx	becomes	∃x[P ∧ Fx]
∃xFx ∧ P	becomes	∃x[Fx ∧ P]
P ∧ ∀xFx	becomes	∀x[P ∧ Fx]
∀xFx ∧ P	becomes	∀x[Fx ∧ P]

If the quantifier is on the front of a consequent of a conditional, move the quantifier to the front of the conditional, having scope over it. This rule is justified by ie/conf.

P → ∃xFx	becomes	∃x[P → Fx]
P → ∀xFx	becomes	∀x[P → Fx]

If the quantifier is on the front of an antecedent of a conditional, move the quantifier to the front of the conditional, having scope over it, changing the quantifier from existential to universal, or vice versa. This rule is justified by ie/conf.

∃xFx → P	becomes	∀x[Fx → P]
∀xFx → P	becomes	∃x[Fx → P]

The moves above are to be made first using a quantifier that is not within the scope of any other quantifier; this quantifier migrates to the very front of the formula. Then any remaining quantifier that is not within the scope of any remaining quantifier migrates to a point just to the right of the previous quantifier, and so on until all quantifiers are moved to the prenex string.

Notice that once biconditionals have been eliminated, the remaining quantifier moves leave the internal molecular structure of the formula intact. For example, if the quantifiers were erased, this would be a conditional with a conjunction as antecedent and a disjunction as consequent.

$$\forall y\exists x[Fx \wedge Gy] \rightarrow \exists z[Hz \vee \forall uS(zu)]$$

When the quantifiers are moved to the front, the internal structure actually is a conditional with a conjunction as antecedent and a disjunction as consequent:

$$\forall y\exists x[Fx \wedge Gy] \rightarrow \exists z[Hz \vee \forall uS(zu)]$$
$$\exists y[\ \exists x[Fx \wedge Gy] \rightarrow \exists z[Hz \vee \forall uS(zu)]\ ]$$
$$\exists y\forall x[\ Fx \wedge Gy \rightarrow \exists z[Hz \vee \forall uS(zu)]\ ]$$
$$\exists y\forall x\ \exists z[\ Fx \wedge Gy \rightarrow [Hz \vee \forall uS(zu)]\ ]$$
$$\exists y\forall x\ \exists z[\ Fx \wedge Gy \rightarrow \forall u[Hz \vee S(zu)]\ ]$$
$$\exists y\forall x\ \exists z\ \forall u\underbrace{[\ Fx\wedge Gy \rightarrow Hz\vee S(zu)}\ ]$$

conditional with a conjunction as antecedent and disjunction as consequent

The process described above may sometimes be applied in more than one way. For example, in

$$\forall xHx \rightarrow \forall yKy$$

neither quantifier is within the scope of the other, and so either one may be moved first. The two options are:

$$\forall xHx \rightarrow \forall yKy \ \Rightarrow\ \exists x[Hx \rightarrow \forall yKy] \ \Rightarrow\ \exists x\forall y[Hx \rightarrow Ky]$$

$$\forall xHx \rightarrow \forall yKy \ \Rightarrow\ \forall y[\exists xHx \rightarrow Ky] \ \Rightarrow\ \forall y\exists x[Hx \rightarrow Ky]$$

The two resulting formulas are not the same; they differ in having an existential and a universal quantifier permuted. Generally such permutation produces nonequivalent formulas, but in this special case they are equivalent. A proof of the equivalence in this case can be produced by giving a biconditional derivation in which one starts with one of the forms, goes back by stages to the original formula, and then moves to the other form, like this:

$\therefore\ \forall y\exists x[Hx \rightarrow Ky] \leftrightarrow \exists x\forall y[Hx \rightarrow Ky]$

1. ~~Show~~ $\forall y\exists x[Hx \rightarrow Ky] \leftrightarrow \exists x\forall y[Hx \rightarrow Ky]$

2.	$\forall y\exists x[Hx \rightarrow Ky]$	ass bd	
3.	$\forall y[\forall xHx \rightarrow Ky]$	2 ie/conf	
4.	$\forall xHx \rightarrow \forall yKy$	3 ie/conf	
5.	$\exists x[Hx \rightarrow \forall yKy]$	4 ie/conf	
6.	$\exists x\forall y[Hx \rightarrow Ky]$	5 ie/conf	bd

EXERCISES

1. The following theorems resemble the last example from the text above. Prove them.

T263 $\forall x\exists y[Fx\rightarrow Gy] \leftrightarrow \exists y\forall x[Fx\rightarrow Gy]$

T264 $\forall x\exists y[Fx\wedge Gy] \leftrightarrow \exists y\forall x[Fx\wedge Gy]$

T265 $\forall x\exists y[Fx\vee Gy] \leftrightarrow \exists y\forall x[Fx\vee Gy]$

T266 $\forall x\forall y\exists z[Fx\wedge Gy\rightarrow Hz] \leftrightarrow \forall y\exists z\forall x[Fx\wedge Gy\rightarrow Hz]$

2. Put each of the following formulas into prenex form. In each case give a biconditional derivation that shows that the prenex form is equivalent to the original formula.

a. $\forall x\exists yP(xy) \rightarrow \exists uP(uu)$
b. $\forall x[\exists uR(ux) \rightarrow \exists uR(xu)]$
c. $\forall x\exists yA(xy) \vee \forall x\exists yA(yx)$
d. $\forall x\forall y[R(xy)\leftrightarrow R(yx)]$

8 SOME THEOREMS

Here are a few theorems that are of special interest.

T249 $\forall x \forall y F(xy) \rightarrow \exists x \exists y F(xy)$ <a generalization of T238>

T253 $\exists x \forall y F(xy) \rightarrow \forall y \exists x F(xy)$ <a "quantifier switch". Note that the quantifiers do not switch in the opposite direction>

T254 $\exists x \exists y F(xy) \leftrightarrow \exists x \exists y[F(xy) \vee F(yx)]$

T256 $\forall x \exists y[F(xy) \wedge Gy] \rightarrow \exists x \exists y[F(xy) \wedge Gx]$

T259 $\forall x \exists z[Fx \rightarrow \exists y[Gy \rightarrow Hz]] \leftrightarrow [\exists x Fx \wedge \forall x Gx \rightarrow \exists x Hx]$

T260 $\forall x[Fx \rightarrow \exists y[Gy \wedge [Hy \vee Hx]]] \leftrightarrow \exists x[Gx \wedge Hx] \vee \sim\exists x Fx \vee [\exists x Gx \wedge \forall x[Fx \rightarrow Hx]]$

T262 $\exists y[\exists x F(xy) \leftrightarrow Gy] \leftrightarrow \exists y \forall x \exists z[[F(xy) \rightarrow Gy] \wedge [Gy \rightarrow F(zy)]]$

The following theorems are of interest in the application in which 'M' stands for set membership; that is, where 'M(①②)' means that ① is a member of the set ②. (We also read this as saying that set ② "contains" ①.) In what is generally called "naïve" set theory, it is assumed that there is a set corresponding to any condition you can express. For example, there is a set, x, whose members are things that are giraffes:

$\exists x \forall z[M(zx) \leftrightarrow Gz]$

There is also a set whose members are all the things that aren't giraffes:

$\exists x \forall z[M(zx) \leftrightarrow \sim Gz]$

Likewise, there is a set whose members are themselves sets which contain at least one giraffe:

$\exists x \forall z[M(zx) \leftrightarrow \exists y[Gy \wedge M(yz)]]$

There is a set that contains every set that contains something that contains something:

$\exists x \forall z[M(zx) \leftrightarrow \exists y \exists u[M(uy) \wedge M(yz)]]$

In the early 1900's, Bertrand Russell considered the set whose members don't contain themselves. Naïve set theory says that there must be a set which contains exactly the sets that do not contain themselves:

$\exists x \forall z[M(zx) \leftrightarrow \sim M(zz)]$

This purported set is called "the Russell set". Russell (among others) showed that there can't be a Russell set. You can show this also. Just construct an easy derivation for this theorem which denies that there is a Russell set:

T269 $\sim\exists y \forall x[M(xy) \leftrightarrow \sim M(xx)]$ <a Russell set does not exist>

A "Russell subset" of a set w is that set which contains all and only those members of w which are not members of themselves. Generally, there is no problem about there being a Russell subset of a set. But one can show that if every set has a Russell subset, then there is no universal set; that is, there is no set which contains everything:

T270 $\forall z \exists y \forall x[M(xy) \leftrightarrow M(xz) \wedge \sim M(xx)] \rightarrow \sim\exists z \forall x M(xz)$ <if "Russell subsets" always exist, there is no universal set>

Here are two more general statements about sets:

T271 $\sim\exists y\forall x[M(xy) \leftrightarrow \sim\exists z[M(xz)\land M(zx)]]$ <there is no set whose members include a thing if and only if it has no member which it is a member of>

T272 $\exists y\forall x[M(xy)\leftrightarrow M(xx)] \rightarrow \sim\forall x\exists y\forall z[M(zy)\leftrightarrow\sim M(zx)]$

<if there is a set of all sets that are members of themselves, not every set x has another set whose members are all the things that are not members of x>

EXERCISES

Prove the theorems above.

9 SHOWING INVALIDITY

The introduction of many-place predicates requires some refinements in our technique for showing predicate calculus invalidity. The fundamental idea remains the same: describe a possible situation in which the premises of an argument of the given form are all true and the conclusion false. But the presence of many-place predicates brings with it two complications, a simple one and a complex one.

The simple complication is due to the fact that when we interpret a many-place predicate there are *combinations* of things in the universe to take into account. For example, consider this argument:

$\forall x\exists y F(xy)$
$\therefore\ \exists y\forall x F(xy)$

Suppose we consider a situation in which exactly three things exist; in particular, our universe is {0, 1, 2}. Previously, for a given predicate we needed to choose for each entity, 0, 1, and 2 whether or not it was in the extension of the predicate. But two-place predicates don't have extensions of that sort; it makes no sense to ask which individual things a two-place predicate is true of. This is because a two-place predicate holds of *pairs* of things. So we need to say which pairs of things are in the extension of each predicate. For example, we might decide that 'F' holds of the following pairs:

$\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,0\rangle$

We indicate these choices by writing:

COUNTER-EXAMPLE

Universe: {0, 1, 2}

F: $\{\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,0\rangle\}$

With these choices the first premise is true, because for each thing in the universe, 'F' relates it to something: 'F' relates 0 to 1, and 1 to 2, and 2 to 0. But the conclusion is false, because there is nothing in the universe such that 'F' relates everything to it. 0 won't do because, for example, 1 isn't related to it, and 1 won't do because, for example, 2 isn't related to it, and 2 won't do because, for example, 0 isn't related to it. This counter-example shows that the argument is not valid in the predicate calculus.

Another example:

$\forall x[Fx \leftrightarrow \exists y R(xy)]$
$Fa \wedge Fb$
$\therefore\ R(ab) \vee R(ba)$

COUNTER-EXAMPLE:

Universe: {0, 1, 2}

a: 0
b: 1
F: {0,1}
R: $\{\langle 0,2\rangle, \langle 1,2\rangle\}$

With these choices the first premise is true because the things that are F are 0 and 1, and the things that bear relation R to something are also 0 and 1. The second premise is true because both 'a' and 'b' stand for things in the extension of 'F'. And the conclusion is false because R does not relate 0 to 1, or 1 to 0.

As in the previous chapter, if it is difficult to assess the truth of sentences in a finite model, one may be able to produce an equivalent truth-functional expansion. Here is an example; suppose that you want to show the invalidity of this argument:

$\forall x\forall y\exists z R(xyz)$
$\therefore\ \forall x\exists z\forall y R(xyz)$

The following counter-example is proposed, in which the three-place relation R has an extension consisting of triples:

Universe: {0,1}

R: $\{\langle 0,0,1\rangle, \langle 0,1,0\rangle, \langle 1,0,1\rangle, \langle 1,1,0\rangle\}$

Then the premise is true and the conclusion false. But this may not be obvious. If so, one may expand the premise and conclusion as follows. Suppose that a_0 stands for 0 and a_1 for 1. Begin with the premise. Eliminating its first (universal) quantifier we get the conjunction:

$$\forall y\exists zR(\mathbf{i}_0yz) \wedge \forall y\exists zR(\mathbf{i}_1yz)$$

Eliminating the next universal quantifier in each conjunct gives a four-part conjunction:

$$\exists zR(\mathbf{i}_0\mathbf{i}_0z) \wedge \exists zR(\mathbf{i}_0\mathbf{i}_1z) \wedge \exists zR(\mathbf{i}_1\mathbf{i}_0z) \wedge \exists zR(\mathbf{i}_1\mathbf{i}_1z)$$

Finally, eliminating the existential quantifiers in each conjunct gives the following sentence:

$$[R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_0) \vee R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_1)] \wedge [R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_0) \vee R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_1)] \wedge [R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_0) \vee R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_1)] \wedge [R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_0) \vee R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_1)]$$

The atomic sentences are now easy to evaluate. For example, we know that '$R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_0)$' is false because <0,0,0> is not in the extension of R. And '$R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_1)$' is true because <0,0,1> is in the extension of R. And so on. Each disjunctive conjunct is true, so the premise is true:

$$[\underbrace{R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_0)}_{F} \vee \underbrace{R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_1)}_{T}]_{T} \wedge [\underbrace{R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_0)}_{T} \vee \underbrace{R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_1)}_{F}]_{T} \wedge [\underbrace{R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_0)}_{F} \vee \underbrace{R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_1)}_{T}]_{T} \wedge [\underbrace{R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_0)}_{T} \vee \underbrace{R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_1)}_{F}]_{T}$$

Regarding the conclusion, eliminating its first (universal) quantifier we get the conjunction:

$$\exists z\forall yR(\mathbf{i}_0yz) \wedge \exists z\forall yR(\mathbf{i}_1yz)$$

Now eliminating the initial existential quantifiers in each conjunct, we get:

$$[\forall yR(\mathbf{i}_0y\mathbf{i}_0) \vee \forall yR(\mathbf{i}_0y\mathbf{i}_1)] \wedge [\forall yR(\mathbf{i}_1y\mathbf{i}_0) \vee \forall yR(\mathbf{i}_1y\mathbf{i}_1)]$$

Finally, eliminating the remaining universal quantifiers we get the following sentence, which is a two-conjunct conjunction each of whose conjuncts is a disjunction of conjuncts.

$$[[R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_0) \wedge R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_0)] \vee [R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_1) \wedge R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_1)]] \wedge [[R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_0) \wedge R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_0)] \vee [R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_1) \wedge R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_1)]]$$

Assessing the atomic sentences as above yields these truth values:

$$[[\underbrace{R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_0)}_{F} \wedge \underbrace{R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_0)}_{T}]_{F} \vee [\underbrace{R(\mathbf{i}_0\mathbf{i}_0\mathbf{i}_1)}_{T} \wedge \underbrace{R(\mathbf{i}_0\mathbf{i}_1\mathbf{i}_1)}_{F}]_{F}] \wedge [[\underbrace{R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_0)}_{F} \wedge \underbrace{R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_0)}_{T}]_{F} \vee [\underbrace{R(\mathbf{i}_1\mathbf{i}_0\mathbf{i}_1)}_{T} \wedge \underbrace{R(\mathbf{i}_1\mathbf{i}_1\mathbf{i}_1)}_{F}]_{F}]$$

The disjuncts are all false, so each conjunct is false, and the conclusion is false.

EXERCISES

1. Give counter-examples to show that each of the following arguments are invalid in the predicate calculus.

a. $\forall x[Fx \rightarrow \exists y[Gy \wedge R(xy)]]$
$\forall x[Gx \rightarrow {\sim}R(xx)]$
$\therefore\ \exists x[Fx \wedge \forall y[Gy \wedge R(xy)]]$

b. $\forall x[F(xa) \leftrightarrow G(xb)]$
$\therefore\ \exists x\exists y[F(xy) \wedge G(xy)]$

c. $\forall x[Hx \rightarrow R(ax)]$
$\forall x\forall y[R(xy) \leftrightarrow R(yx)]$
$\therefore\ \exists y\forall xR(xy)$

d. $\forall x[Fx \rightarrow \exists yR(xy)]$
$\forall x[{\sim}Fx \rightarrow \exists yR(yx)]$
$\therefore\ \forall x\exists y[R(xy) \wedge R(yx)]$

e. $\forall x\forall y[F(xy) \leftrightarrow \exists z[G(xz) \wedge {\sim}G(yz)]]$
$\forall x\forall y[F(xy) \rightarrow F(yx)]$
$\therefore\ \forall x\forall y[G(xy) \rightarrow G(yx)]$

10 INFINITE UNIVERSES

The second complication to our technique arises only in some cases. It has to do with infinite universes. Consider the following argument:

$\forall x \exists y R(xy)$
$\forall x \forall y \forall z[R(xy) \wedge R(yz) \rightarrow R(xz)]$
$\therefore \exists x R(xx)$

The first premise says that each thing is related by R to something. The second says that relation R is *transitive*: if something is related by R to something else, and that something else is related to a further thing, the first thing must be related to this further thing. Finally, the conclusion says that something is related by R to itself.

This argument is invalid, but this cannot be shown using a counter-example with a finite universe. Instead, we have to devise a counter-example using an infinite universe.

Here is why a finite universe will not work. Consider an attempt to create a counter-example using a finite universe. Let us start with the smallest choice: there is only 0. Now by the first premise, 'R' relates 0 to something. Since 0 is all there is, in order to make the first premise true, 'R' must relate 0 to 0. But then the conclusion will be true. So a one-element universe won't do.

OK, let's try two things, 0 and 1. Again, 'R' must relate 0 to something. It can't relate 0 to 0, as we saw above. So 'R' must relate 0 to 1. Looking at the first premise again, 'R' must relate 1 to something. It can't relate 1 to 1, for the same reason as before; making the conclusion false forbids anything being related to itself. So 'R' must relate 1 to 0. Fine. But now the second premise comes into play. The second premise says that 'R' is *transitive*: if it relates one thing to a second, and that second to a third, it relates the first to the third. But it does relate one thing (0) to a second thing (1), and it relates that second thing (1) to a third thing (0), so it must now relate the first (0) to the third (0). Which is ruled out by the third premise.

> (You might think that this reasoning doesn't work, since we have talked about a "first" thing and a "second" thing, and a "third" thing. And in the application we used, we made 0 be the "third" thing. But there are only two of them: 0 and 1, and so it seems wrong to talk of a *third* thing.
>
> The answer to this objection is that this use of 'first', 'second', and 'third' is just a manner of speaking that is used in natural language to keep track, not of three things, but of three variables. There aren't any variables in English so we speak in this way. This usage can be avoided if we argue as follows:
>
>> "The second premise says that 'R' is *transitive*: if it relates one thing, x, to a thing, y, and it relates thing, y, to a thing, z, then it relates thing x to thing z. But it relates one thing, 0, to 1, and it relates 1 to 0, so it must now relate 0 to 0. Which is ruled out by the required falsehood of the conclusion.")

Trying three things won't work either. The premises require that 0 is related to 1, and 1 to something else, 2, but then 2 must be related to something. Not to itself, because we need the conclusion to be false, and not to either 0 or 1, because the reasoning given above reapplies. And so on. These premises require that each thing is related to something *new*, and so on *ad infinitum* ("to infinity").

We therefore need to consider a situation in which there are an infinite number of things, say, all of the integers {0, 1, 2, . . }. Previously we gave the extensions of predicates by listing the things, or the pairs of things, in their extensions. But if the extensions happen to be infinite, their members can't be given in a finite list. Instead, we need to explain in words how things are related by each predicate. We can do this by giving a scheme of abbreviation. Here is a way to do this, describing a situation in which the premises are all true and the conclusion false:

COUNTER-EXAMPLE:

> Universe = the non-negative integers: {0, 1, 2, 3, . . }
>
> R(①②) holds when ①<②

That is, 'R' relates any two things, here called '①' and '②', if and only if the first thing, ①, is arithmetically less than the second thing, ②.

Now consider how the parts of the argument fare in this counter-example.

	∀x∃yR(xy)	True: Every integer is less than some integer
	∀x∀y∀z[R(xy) ∧ R(yz) → R(xz)]	True: For any integers, x, y, z, if x is less than y and y is less than z then x is less than z
∴	∃xR(xx)	False: No integer is less than itself

Here is another invalid argument:

∀x∀y[R(xy) → ~R(yx)]
∀x∃yR(yx)
∀x∀y∀z[R(xy) ∧ R(yz) → R(xz)]
∴ ∃x∀yR(xy)

Again, a finite universe will not work to produce a counter-example. (Try it and see.) But a counter-example with an infinite universe is possible. For example:

COUNTER-EXAMPLE

Universe: {0, 1, 2, . . . }

R(①②) holds when ①>②

The first premise is true in this counter-example because whenever one thing is greater than another, that other is not greater than the first. The second premise says that for anything there is something greater than it, which is true in our infinite universe. The third is the transitivity condition again, which holds for greater-thanness. And the conclusion is false because there isn't a thing in this universe which is greater than everything.

In constructing counter-examples in this way one must keep in mind that each name must be assigned something that is actually in the chosen universe.

~∃xR(xx)
∀x∃yR(xy)
∀x∀y∀z[R(xy) ∧ R(yz) → R(xz)]
∴ ∃xR(xa)

COUNTER-EXAMPLE

Universe: {1, 2, . . . }

a: 1

R(①②) holds when ①<②

The first premise is true because nothing in the given universe is less than itself. Ths second is true because for each thing in the universe, there's something that it's less than. The third premise is true because less than is transitive. And the conclusion is false because there isn't a thing in the universe that's less than 1. Of course, there are things that are less than 1, but not in the given universe. Notice that given the universe we have chosen, we could not specify that 'a' stands for 0, because 0 isn't in the given universe. (Of course, we could choose a different universe, say: {0, 1, 2, . . . }, and then we could let 'a' stand for 0.)

EXERCISES

Give counterexamples with infinite domains to show that each of the following arguments is invalid.

a. $\forall x{\sim}R(xx)$
$\forall x\exists yR(xy)$
$\therefore\ \exists x\exists y\exists z[R(xy) \wedge R(yz) \wedge {\sim}R(xz)]$

b. $\forall x\forall y\forall z[R(xy) \wedge R(yz) \rightarrow R(xz)]$
${\sim}\exists x[Ex \wedge R(xx)]$
$\forall x\exists y[Oy \wedge R(xy)]$
$\therefore\ \exists x{\sim}[Ox \leftrightarrow Ex]$

c. $\forall x[Ex \rightarrow \exists y[Fy \wedge S(yx)]]$
$\forall x[Fx \rightarrow \exists y[Ey \wedge S(yx)]]$
$\forall x[Ex \vee Fx]$
$\forall x\forall y\forall z[S(xy) \wedge S(yz) \rightarrow S(xz)]$
$\therefore\ \exists xS(xx)$

Answers to the Exercises -- Chapter 4

SECTION 1

1. Which of the following are formulas in official notation? Which are formulas in informal notation? Which are not formulas at all?

a.	~~F(xa)	Formula – official notation
b.	[∀xG(bx) → ~∃yG(yx)]	Formula – official notation
c.	∀xG(bx) → ~∃yG(yx)	Formula – informal notation
d.	~Fa & ~G(aa) & ~Fb & Gxb	Not a formula -- lacks parentheses around 'xb'
e.	~F(a) ∨ ~G(ab)	Not a formula -- parentheses not used with 1-place predicates
f.	~Fa ∨ ~Gab	Not a formula – lacks parentheses around 'ab'
g.	~∃x[~Fx → ∀yG(yy)]	Formula –official notation
h.	∃x∀y~Fxy	Not a formula – lacks parentheses around 'xy'
i.	∃x∃yF[xy]	Not a formula –parentheses required; not brackets

2 SYMBOLIZING SENTENCES USING MANY-PLACE PREDICATES

1. Symbolize each of the following:

a. *Hans sees every doctor but Amanda doesn't see any doctor.*
∀x[Dx → Hans sees x] but ~∃x[Dx ∧ Amanda sees x]
∀x[Dx → S(hx)] ∧ ~∃x[Dx ∧ S(ax)]

b. *Hans, who owns a dog, doesn't own a cat.*
Hans owns a dog ∧ ~ Hans owns a cat
∃x[Dx ∧ O(hx)] ∧ ~∃x[Cx ∧ O(hx)]

c. *Hans loves Amanda but she doesn't love him.*
L(ha) ∧ ~L(ah)

d. *Neither Hans nor Amanda has a cat.*
~Hans has a cat ∧ ~ Amanda has a cat
~∃x[Cx ∧ H(hx)] ∧ ~∃x[Cx ∧ H(ax)]

f. *Some hyena and some giraffe like each other*.
∃x∃y[Hx ∧ Gy ∧ x and y like each other]
∃x∃y[Hx ∧ Gy ∧ L(xy) ∧ L(yx)]

g. *Some giraffe likes every baboon.*
∃x[Gx ∧ x likes every baboon]
∃x[Gx ∧ ∀y[By → L(xy)]]

h. *Some giraffe that likes every baboon likes no hyena.*
∃x[x is a giraffe that likes every baboon ∧ x likes no hyena]
∃x[Gx ∧ x likes every baboon ∧ x likes no hyena]
∃x[Gx ∧ ∀y[By → L(xy)] ∧ ~∃z[Hz ∧ L(xz)]]

i. *Some giraffe likes every baboon that likes no hyena*
∃x[Gx ∧ x likes every baboon that likes no hyena]
∃x[Gx ∧ ∀y[y is a baboon that likes no hyena → x likes y]]
∃x[Gx ∧ ∀y[By ∧ ~∃z[Hz ∧ L(yz)] → L(xy)]]

j. Some giraffe likes every baboon that likes it
∃x[Gx ∧ ∀y[y is a baboon that likes x → x likes y]]
∃x[Gx ∧ ∀y[By ∧ L(yx) → L(xy)]]

k. *Eileen resides in a big city*. <use 'R(①②)' for '① resides in ②'>
∃x[x is a big city ∧ Eileen resides in x]
∃x[Bx ∧ Cx ∧ R(ex)]

l. *Eileen and Betty both reside in the same city*.
∃x[x is a city ∧ Eileen resides in x ∧ Betty resides in x]
∃x[Cx ∧ R(ex) ∧ R(bx)]

m. *If Hank resides in Brea then he attends UCLA; otherwise he doesn't attend UCLA.*
[Hank resides in Brea → Hank attends UCLA] ∧ [Hank doesn't reside in Brea → Hank doesn't attend UCLA]
[R(hb) → A(ha)] ∧ [~R(hb) → ~A(ha)]

n. *If David and Hank both live in Brea then David attends a private school and Hank attends a public school.*
D: private E: public C: school L(①②): ① lives in ② A(①②): ① attends ②
David lives in Brea ∧ Hank lives in Brea→ ∃x[x is a private school ∧ David attends x] ∧ ∃x[x is a public school ∧ Hank attends x]
L(db) ∧ L(hb) → ∃x[Dx ∧ Cx ∧ A(dx)] ∧ ∃x[Ex ∧ Cx ∧ A(hx)]

o. *Nobody who comes from Germany attends a Californian school.* F: Californian
~∃x[x comes from Germany ∧ x attends a Californian school]
~∃x[x comes from Germany ∧ ∃y[y is a Californian school ∧ x attends y]]
~∃x[C(xg) ∧ ∃y[Fy ∧ Cy ∧ A(xy)]]

p. No giraffe likes Fido unless it is crazy
∀x[x is a giraffe → x doesn't like Fido unless x is crazy]
∀x[Gx → ~L(xf) ∨ Cx]
or
~∃x[x is a giraffe ∧ x likes Fido ∧ x isn't crazy]
~∃x[Gx ∧ L(xf) ∧ ~Cx]

q. Nobody gives a book to a freshman unless it is inexpensive G(①②③): ① gives ② to ③
∀x∀y[x is a person ∧ y is a book → x doesn't give y to a freshman unless y is inexpensive]
∀x∀y[Ex ∧ By → ~∃z[Fz∧G(xyz)] ∨ Iy]
or
~∃x∃y[Ex ∧ By ∧ ∃z[Fz∧G(xyz)] ∧ ~Iy]

3 DERIVATIONS

Show each of the following arguments to be valid.

1. ∀x∀y∀z[S(xy) ∧ S(yz) → S(xz)]
S(bc) ∧ S(ab)
∴ S(ac)

1. Show S(ac)

2.	S(ab) ∧ S(bc) → S(ac)	pr1 ui ui ui
3.	S(ab)	pr2 s
4.	S(bc)	pr2 s
5.	S(ac)	3 4 adj 2 mp dd

2. ∀x∀y[Ax ∧ By → [S(xy) ↔ ~S(yx)]]
∴ ∀x∀y[Ax ∧ By → [S(xy) ∨ S(yx)]]

1.	~~Show~~ ∀x∀y[Ax ∧ By → [S(xy) ∨ S(yx)]]	
2.	~~Show~~ ∀y[Ax ∧ By → [S(xy) ∨ S(yx)]]	
3.	~~Show~~ Ax ∧ By → [S(xy) ∨ S(yx)]	
4.	Ax ∧ By	ass cd
4.	Ax ∧ By → [S(xy) ↔ ~S(yx)]	pr1 ui ui
5.	S(xy) ↔ ~S(yx)	3 4 mp
6.	~S(yx) → S(xy)	5 bc
7.	~~Show~~ S(xy) ∨ S(yx)	
8.	~[S(xy) ∨ S(yx)]	ass ud
9.	~S(xy) ∧ ~S(yx)	8 dm
10.	~S(yx)	9 s
11.	S(xy)	6 10 mp
12.	~S(xy)	9 s 11 id
13.		7 cd
14.		3 ud
15.		2 ud

3. ∀x∃yS(xy)
∀x∀y[Cx ∧ S(xy) → Dy]
∀x∀y[Dx ∧ S(yx) → Dy]
∴ ~∃x[Cx ∧ ~Dx]

1.	~~Show~~ ~∃x[Cx ∧ ~Dx]	
2.	∃x[Cx ∧ ~Dx]	ass id
3.	Cu ∧ ~Du	2 ei
4.	∃yS(uy)	pr1 ui
5.	S(uv)	4 ei
6.	Cu ∧ S(uv) → Dv	pr2 ui ui
7.	Dv	3 s 5 adj 6 mp
8.	Dv ∧ S(uv) → Du	pr3 ui ui
9.	Du	7 5 adj 8 mp
10.	~Du	3 s 9 id

4. ∃xEx ∧ ∃x~Ex
∀x∀y[Ex ∧ S(xy) → Ey]
∴ ∃x∃y~S(xy)

1.	~~Show~~ ∃x∃y~S(xy)	
2.	∃xEx	pr1 s
3.	Eu	2 ei
4.	∃x~Ex	pr1 s
5.	~Ev	4 ei
6.	Eu ∧ S(uv) → Ev	pr2 ui ui
7.	~[Eu ∧ S(uv)]	5 6 mt
8.	~Eu ∨ ~S(uv)	7 dm
9.	~S(uv)	3 dn 8 mtp
10.	∃x∃y~S(xy)	9 eg eg dd

5. ∀x∀y[S(xy) ↔ S(yx)]
∃x∃y[Ax ∧ By ∧ S(xy)]
∴ ∃x∃y[By ∧ Ax ∧ S(yx)]]

1. ~~Show~~ ∃x∃y[By ∧ Ax ∧ S(xy)]]

2.	Au ∧ Bv ∧ S(uv)	pr2 ei ei
3.	S(uv)	2 s
4.	S(uv) ↔ S(vu)	pr1 ui ui
5.	S(vu)	4 bc 3 mp
6.	Au	2 s s
7.	Bv	2 s s
8.	Bv ∧ Au ∧ S(vu)	7 6 adj 5 adj
9.	∃x∃y[By ∧ Ax ∧ S(yx)]	8 eg eg dd

6. ∃x[Ax ∧ ∀y[By → S(xy)]]
∀x∀y[Bx ↔ Cy]
∴ ∀x[Cx → ∃yS(yx)]

1.~~Show~~ ∀x[Cx → ∃yS(yx)]

2.	~~Show~~ Cx → ∃yS(yx)	
3.	Cx	ass cd
4.	Au ∧ ∀y[By → S(uy)]	pr1 ei
5.	∀y[By → S(uy)]	4 s
6.	Bx → S(ux)	5 ui
7.	Bx ↔ Cx	pr2 ui ui
8.	Bx	7 bc 3 mp
9.	S(ux)	6 8 mp
10.	∃yS(yx)	9 eg cd
11.		2 ud

7. Answers are not supplied for derivations of numbered theorems.

4 THE RULE "INTERCHANGE OF EQUIVALENTS"

1. P ↔ Q ∨ R
   ~Q → ~S ∨ P
   ∴ R ∨ ~Q

1.	Show R ∨ ~Q		
2.	~~P ↔ Q ∨ R	pr1 ie/dn	OK
3.	~~P ↔ P	2 ie/pr2	no -- appeals to wrong premise
4.	~~P	3 ie/dn	no -- does not result from 3 by interchanging an equivalent part
5.	P	4 ie/dn	OK
6.	~S ∨ P	5 add	
7.	~Q	6 ie/pr2	no -- premise 2 is not a biconditional
8.	R ∨ ~Q	7 add dd	

2. ∀x∃y[Ax ↔ R(xy)]
   ∀z∀y[R(zy) ↔ S(yz)]
   ∀x[[Ax↔Ax] ↔ Ax]
   ∴ Au

1.	Show ∃x[R(xx) ∧ ~Ax]		
2.	∃y[Ax ↔ R(xy)]	pr1 ui	
3.	Au ↔ R(xu)	2 ie	
4.	R(xu) ↔ S(ux)	pr2 ui	
5.	Au ↔ S(ux)	4 ie/3	OK
6.	Au ↔ S(ux)	3 ie/4	OK
7.	Au ↔ Au	5 ie/6	OK
8.	[Au↔Au] ↔ Au	pr3 ui	
9.	Au	7 ie/8 dd	OK -- results from 7 by changing 'Au↔Au' to 'Au', which is justified by line 8

5 BICONDITIONAL DERIVATIONS

1. Prove the given biconditional without using a biconditional derivation and also without using the rule for interchange of equivalents:

∴ ~P ∧ ~~Q ↔ ~[P ∨ ~Q]

1.	Show ~P ∧ ~~Q ↔ ~[P ∨ ~Q]	
2.	Show ~P ∧ ~~Q → ~[P ∨ ~Q]	
3.	~P ∧ ~~Q	ass cd
4.	Show ~[P ∨ ~Q]	
5.	P ∨ ~Q	ass id
6.	~Q	3 s 5 mtp
7.	~~Q	3 s 6 id
8.	Show ~[P ∨ ~Q] → ~P ∧ ~~Q	
9.	~[P ∨ ~Q]	ass cd
10.	~P ∧ ~~Q	9 dm cd
11.	~P ∧ ~~Q → ~[P ∨ ~Q]	2 8 cb dd

2. Derivations are not given here for numbered theorems.

6 SENTENCES WITHOUT OVERLAY OF QUANTIFIERS

For each of the following formulas, find an equivalent formula which has no overlay of quantifiers, and prove that it is equivalent.

a. ∀z[∃u[Fu → Gz] → Fz]

1. ~~Show~~ ∀z[∃u[Fu → Gz] → Fz] ↔ [[∀uFu → Gz] → ∀zFz]

2.	∀z[∃u[Fu → Gz] → Fz]	ass bd
3.	∃u[Fu → Gz] → ∀zFz	2 ie/conf
4.	[∀uFu → Gz] → ∀zFz	3 ie/conf
5.		4 bd

b. ∃z∀x[Fx ↔ Fz]

One way to do this kind of problem is to ask yourself how to express what a formula says in terms of a formula without overlay, and then prove that that formula is equivalent to the original. Here is an example, that leads to a long derivation.

1.	~~Show~~ ∃z∀x[Fx ↔ Fz] ↔ ∀xFx ∨ ~∃xFx	
2.	~~Show~~ ∃z∀x[Fx ↔ Fz] → ∀xFx ∨ ~∃xFx	
3.	∃z∀x[[Fx → Fz]	ass cd
4.	~~Show~~ ∀xFx ∨ ~∃xFx	
5.	~[∀xFx ∨ ~∃xFx]	ass id
6.	~∀xFx ∧ ~~∃xFx	5 dm
7.	~∀xFx	6 s
8.	∃x~Fx	7 qn
9.	~Fu	8 ei
10.	∃xFx	6 s dn
11.	Fv	10 ei
12.	∀x[Fx ↔ Fw]	3 ei
13.	Fv ↔ Fw	12 ui
14.	Fw	13 bc 11 mp
15.	Fu ↔ Fw	12 ui
16.	Fu	15 bc 14 mp
17.		9 16 id
18.		4 cd
19.	~~Show~~ ∀xFx ∨ ~∃xFx → ∃z∀x[Fx ↔ Fz]	
20.	∀xFx ∨ ~∃xFx	ass cd
21.	~~Show~~ ∃z∀x[Fx ↔ Fz]	
22.	~∃z∀x[Fx ↔ Fz]	ass id
23.	∀z~∀x[Fx ↔ Fz]	22 qn
24.	~∀x[Fx ↔ Fz]	23 ui
25.	∃x~[Fx ↔ Fz]	24 qn
26.	~[Fu ↔ Fz]	25 ei
27.	Fu ↔ ~Fz	26 nb
28.	~~Show~~ ~∀xFx	
29.	∀xFx	ass id
30.	Fu	29 ui
31.	~Fz	27 bc 30 mp
32.	Fz	29 ui 31 id
33.	~∃xFx	28 20 mtp
34.	∀x~Fx	33 qn
35.	~Fz	34 ui
36.	Fu	27 bc 35 mp
37.	~Fu	34 ui 36 id
38.		21 cd
39.	∃z∀x[Fx ↔ Fz] ↔ ∀xFx ∨ ~∃xFx 2 19 cb	dd

Another way to do it is to just go through and change parts to their equivalents. This is convenient if you make use of derived rules, beginning by setting up the derivation before you know what the final formula will be. Your goal will be to turn subformulas into disjunctions and conjunctions and use the laws:

assoc	associativity
com	commutativity
dist	distribution
conf	confinement
qdist	quantifier distribution
bex	biconditional expansion

1. ~~Show~~ ∃z∀x[Ax ↔ Az] ↔ ?????

2.	∃z∀x[Ax ↔ Az]	ass bd	
3.	∃z∀x[[Ax → Az] ∧ [Az → Ax]]	2 ie/bex	"biconditional expansion"
4.	∃z[∀x[Ax → Az] ∧ ∀x[Az → Ax]]	3 ie/qdist	"quantifier distribution"
5.	∃z[[∃xAx → Az] ∧ [Az → ∀xAx]]	4 ie/conf ie/conf	
6.	∃z[[~∃xAx ∨ Az] ∧ [~Az ∨ ∀xAx]]	5 ie/cdj ie/cdj	
7.	∃z[(~∃xAx∧~Az) ∨ (~∃xAx∧∀xAx) ∨ (Az∧~Az) ∨ (Az∧∀xAx)]	6 ie/dist	
8.	∃z((~∃xAx∧~Az) ∨ (~∃xAx∧∀xAx) ∨ (Az∧~Az)) ∨ ∃z(Az∧∀xAx)	7 ie/qdist	
9.	∃z(~∃xAx∧~Az) ∨ ∃z(~∃xAx∧∀xAx) ∨ ∃z(Az∧~Az) ∨ ∃z(Az∧∀xAx)	8 ie/qdist	
10.	(~∃xAx∧∃z~Az) ∨ ∃z(~∃xAx∧∀xAx) ∨ ∃z(Az∧~Az) ∨ ∃z(Az∧∀xAx)	9 ie/conf	
11.	(~∃xAx∧∃z~Az) ∨ ∃z(~∃xAx∧∀xAx) ∨ ∃z(Az∧~Az) ∨ (∃zAz∧∀xAx)	10 ie/conf	
12.	(~∃xAx∧∃z~Az) ∨ (~∃xAx∧∀xAx) ∨ ∃z(Az∧~Az) ∨ (∃zAz∧∀xAx)	11 ie/vac	

One may then replace the question marks with the formula on line 12, and then add:

13. 12 bd

and box and cancel the original 'show'.

c. ∃xFx ∨ ∀y~Gy This sentence already lacks quantifier overlays.

d. ∃x[∃x[Fx ↔ Gx] → [Fx ↔ Gx]]

1. ~~Show~~ ∃x[∃x[Fx ↔ Gx] → [Fx ↔ Gx]] ↔ [∃x[Fx ↔ Gx] → ∃x[Fx ↔ Gx]]

2.	∃x[∃x[Fx ↔ Gx] → [Fx ↔ Gx]]	ass bd
3.	∃x[Fx ↔ Gx] → ∃x[Fx ↔ Gx]	2 ie/conf bd

e. ∀x∃y∀z[Fx ∧ Gz → Fz ∨ Gy]

The strategy here is to manipulate the parts to get the two atomic formulas containing 'z' together as a unit (lines 3-5) and then apply the confinement laws.

1. ~~Show~~ ∀x∃y∀z[Fx ∧ Gz → Fz ∨ Gy] ↔ ∃xFx → [∀z[~Gz ∨ Fz] ∨ ∃yGy]

2.	∀x∃y∀z[Fx ∧ Gz → Fz ∨ Gy]	ass bd
3.	∀x∃y∀z[Fx → [Gz → Fz ∨ Gy]]	2 ie/exp
4.	∀x∃y∀z[Fx → [~Gz ∨ [Fz ∨ Gy]]]	3 ie/cdj
5.	∀x∃y∀z[Fx → [[~Gz ∨ Fz] ∨ Gy]]	4 ie/assoc
6.	∀x∃y[Fx → ∀z[[~Gz ∨ Fz] ∨ Gy]]	5 ie/conf
7.	∀x∃y[Fx → [∀z[~Gz ∨ Fz] ∨ Gy]]	6 ie/conf
8.	∀x[Fx → ∃y[∀z[~Gz ∨ Fz] ∨ Gy]]	7 ie/conf
9.	∀x[Fx → [∀z[~Gz ∨ Fz] ∨ ∃yGy]]	8 ie/conf
10.	∃xFx → [∀z[~Gz ∨ Fz] ∨ ∃yGy]	9 ie/conf bd

7 PRENEX NORMAL FORMS

1. Derivations are not given here for numbered theorems.

2. Put each of the following formulas into prenex normal form. In each case give a biconditional derivation that shows that the prenex form is equivalent to the original formula.

a. ∀x∃yP(xy) → ∃uP(uu)

Begin by setting up a biconditional derivation. You won't know at the beginning exactly what form the right-hand side of the biconditional will take. So just leave a place for it:

1.	Show [∀x∃yP(xy) → ∃uP(uu)] ↔ ?????	
2.	[∀x∃yP(xy) → ∃uP(uu)]	ass bd

Then carry out the series of equivalences:

1.	Show [∀x∃yP(xy) → ∃uP(uu)] ↔ ?????	
2.	∀x∃yP(xy) → ∃uP(uu)	ass bd
3.	∃x[∃yP(xy) → ∃uP(uu)]	2 ie/conf
4.	∃x∀y[P(xy) → ∃uP(uu)]	3 ie/conf
5.	∃x∀y∃u[P(xy) → P(uu)]	4 ie/conf

Finally fill in the right-hand side of the top biconditional with what you have shown on line 5, and box and cancel:

1.	~~Show~~ [∀x∃yP(xy) → ∃uP(uu)] ↔ ∃x∀y∃u[P(xy) → P(uu)]	
2.	∀x∃yP(xy) → ∃uP(uu)	ass bd
3.	∃x[∃yP(xy) → ∃uP(uu)]	2 ie/conf
4.	∃x∀y[P(xy) → ∃uP(uu)]	3 ie/conf
5.	∃x∀y∃u[P(xy) → P(uu)]	4 ie/conf bd

Derivations for the other examples will be generated in this way: set up a biconditional derivation and then carry it out. Only the final derivations are given below:

b. ∀x[∃uR(ux) → ∃uR(xu)]

The trick here is to use rule av to change bound variables so that the confinement rules will apply.

1.	~~Show~~ ∀x[∃uR(ux) → ∃uR(xu)] ↔ ∀x∀y∃u[R(yx) → R(xu)]	
2.	∀x[∃uR(ux) → ∃uR(xu)]	ass bd
3.	∀x[∃yR(yx) → ∃uR(xu)]	2 ie/av
4.	∀x∀y[R(yx) → ∃uR(xu)]	3 ie/conf
5.	∀x∀y∃u[R(yx) → R(xu)]	4 ie/conf bd

c. ∀x∃yA(xy) ∨ ∀x∃yA(yx)

1.	~~Show~~ [∀x∃yA(xy) ∨ ∀x∃yA(yx)] ↔ ∀x∃y∀u∃v[A(xy) ∨ A(vu)]	
2.	∀x∃yA(xy) ∨ ∀x∃yA(yx)	ass bd
3.	∀x∃yA(xy) ∨ ∀x∃vA(vx)	2 ei/av
4.	∀x∃yA(xy) ∨ ∀u∃vA(vu)	3 ei/av
5.	∀x[∃yA(xy) ∨ ∀u∃vA(vu)]	4 ei/conf
6.	∀x∃y[A(xy) ∨ ∀u∃vA(vu)]	5 ei/conf
7.	∀x∃y∀u[A(xy) ∨ ∃vA(vu)]	6 ei/conf
8.	∀x∃y∀u∃v[A(xy) ∨ A(vu)]	7 ei/conf bd

d. ∀x∀y[R(xy)↔R(yx)]

This is already in prenex normal form.

8 SOME THEOREMS

Derivations are not given here for numbered theorems.

9 SHOWING INVALIDITY

1. Give counter-examples to show that these arguments are invalid in the predicate calculus.

a. $\forall x[Fx \rightarrow \exists y[Gy \wedge R(xy)]]$
$\forall x[Gx \rightarrow \sim R(xx)]$
$\therefore \exists x[Fx \wedge \forall y[Gy \wedge R(xy)]]$

Universe: {0}

F: {}
G: <any choice will do>
R: <any choice will do>

Both premises are true because they contain conditionals with false antecedents for any value of 'x'; the conclusion is false because nothing is F.

Another answer:

Universe: {0,1}

F: {0,1}
G: {0,1}
R: {<1,0>, <0,1>}

The first premise is true because for any choice of 'x' (either 0 or 1) there is something which is G and related to the choice of 'x' by R. The second premise is true because nothing is related to itself by R. The conclusion is false because whatever you pick for 'x' the universal quantifier '$\forall$y' will require that that thing be related to itself by R.

b. $\forall x[F(xa) \leftrightarrow G(xb)]$
$\therefore \exists x \exists y[F(xy) \wedge G(xy)]$

Universe: {0,1}

a: 1
b: 0
F: {<0,1>}
G: {<0,0>}

The premise is true because its instances are all true; choosing 0 for 'x' both sides of the biconditional are true; choosing 1 for 'x' both sides of the biconditional are false. The conclusion is false since there is nothing you can pick for 'x' and 'y' which give you a pair of things that is in the extensions of both F and G.

c. $\forall x[Hx \rightarrow R(ax)]$
$\forall x \forall y[R(xy) \leftrightarrow R(yx)]$
$\therefore \exists y \forall x R(xy)$

Universe: {0,1}

a: 0
H: {0}
R: {<0,0>}

The first premise is true because there is only one thing that is H, and that is 0, and it is related to the thing that 'a' stands for (namely, 0) by R. The second premise says that any pair of things that are related by R are also related in reverse order; the only thing that R applies to is the pair <0,0>, and reversing it makes no difference. The conclusion is false since there isn't anything that is related to everything by R.

d. $\forall x[Fx \rightarrow \exists yR(xy)]$
$\forall x[\sim Fx \rightarrow \exists yR(yx)]$
$\therefore$ $\forall x \exists y[R(xy) \wedge R(yx)]$

Universe: {0,1}

F: {0}
R: {<0,1>}

The first premise is true because whatever is F, namely, 0, is related to something (namely, 1) by R. The second premise is true because for whatever isn't F, namely, 1, something (namely, 0) is related to it by R. The conclusion is false since it says that everything is related to something by R in both directions, and nothing is related to anything by R in both directions.

e. $\forall x \forall y[F(xy) \leftrightarrow \exists z[G(xz) \wedge \sim G(yz)]]$
$\forall x \forall y[F(xy) \rightarrow F(yx)]$
$\therefore$ $\forall x \forall y[G(xy) \rightarrow G(yx)]$

Universe: {0,1,2}

F: {<0,1>, <1,0>, <1,2>, <2,1>, <2,0>, <0,2>}
G: {<0,1>, <1,2>, <2,0>}

The first premise is true since it comes out true for all choices of 'x' and 'y'. (There are nine choices in all; each can be checked on its own.) The second premise says that F is symmetric; this is clearly true since for every pair in the extension of F the reverse pair is also there. The conclusion says falsely that G is symmetric; G holds of <2,0> but not of <0,2>.

Another answer:

Universe: {0,1}

F: {}
G: {<0,1>, <1,1>}

The first premise is true since both sides of the biconditional are false for any choices of 'x' and 'y'; this is clear for the left-hand side since F is true of no pairs at all; the other side can be checked by cases. The second premise is vacuously true. The conclusion falsely says that G is symmetric; but G holds of <0,1> and not of <1,0>.

10 INFINITE UNIVERSES

Give counterexamples with infinite domains to show that each of the following arguments is invalid.

a. $\forall x{\sim}R(xx)$

$\forall x\exists yR(xy)$

$\therefore\ \exists x\exists y\exists z[R(xy) \wedge R(yz) \wedge {\sim}R(xz)]$

Universe: {0, 1, 2, . . . } zero and all the positive integers

R(①②) : ①<②

The first premise says truly that nothing is less than itself. The second says truly that for every non-negative integer there is another non-negative integer that it is less than. The conclusion is false since less than is transitive.

b. $\forall x\forall y\forall z[R(xy) \wedge R(yz) \rightarrow R(xz)]$

${\sim}\exists x[Ex \wedge R(xx)]$

$\forall x\exists y[Oy \wedge R(xy)]$

$\therefore\ \exists x{\sim}[Ox \leftrightarrow Ex]$

Universe: {0, 1, 2, . . . } zero and all the positive integers

E① : ① is even
O① : ① is even
R(①②) : ①<②

The first premise is true since less than is transitive. The second is true since no even number is less than itself. The third premise says truly that for every non-negative integer there is an even non-negative integer that it is less than. The conclusion says falsely that for some non-negative integer it isn't the case that it's even if and only if it's even.

c. $\forall x[Ex \rightarrow \exists y[Fy \wedge S(yx)]]$

$\forall x[Fx \rightarrow \exists y[Ey \wedge S(yx)]]$

$\forall x[Ex \vee Fx]$

$\forall x\forall y\forall z[S(xy) \wedge S(yz) \rightarrow S(xz)]$

$\therefore\ \exists xS(xx)$

Universe: {0, 1, 2, . . . } zero and all the positive integers

E① : ① is even
F① : ① is odd
S(①②) : ①>②

The first premise is true since for every even non-negative integer there is an odd non-negative integer that is greater than it. The second premise is true since for every odd non-negative integer there is an even non-negative integer that is greater than it. The third premise says truly that every non-negative integer is either even or odd. <if you think that 0 is neither even nor odd, then just change the interpretation of 'E' to 'x is even or x=0'> The fourth premise says truly that greater than is transitive. The conclusion says falsely that some non-negative integer is greater than itself.