Peter Smith: An Introduction to Formal Logic (2nd edition)

Appendix on Quantifier Truth Trees

Ideally students should know about both natural deduction and trees early in their logical education. But an introductory text covering both would begin to look rather dauntingly long. So you have to choose, on a balance of considerations for and against. *IFL2* chooses natural deduction.

I still think there are good reasons on the side of starting with truth trees. So you might prefer to follow Chapters 1 to 19 of *IFL2* by looking at the chapters on propositional truth trees at https://www.logicmatters.net/ifl/other-materials/. Then you can read Chapters 25 to 30, 35, 36, 38 to 40 of *IFL2*, followed by these three chapters on truth trees for quantified logic.

These chapters are simply reproduced from *IFL1*. And so the aswers to exercises are available from the *IFL1* webpages.

There's a (relatively) minor difference between the approaches of *IFL1* and *IFL2* which you will need to negotiate. Both texts use the likes of 'm' and 'n' as proper names (with a fixed reference in context), while 'a' and 'b' are recruited as 'arbitrary' names in *IFL1* and 'dummy' names in *IFL2*. The second edition, however, is much sterner about treating these as syntactically different types of expression to mark their different semantic role. The first edition has one category of names (in other word, it adopts policy (B) of *IFL2* §32.1). But this difference should cause you no trouble.

25

Introducing QL trees

Our discussion of the language **QL** has so far been fairly informal and introductory. We'll need to nail down the syntax and semantics much more carefully, beginning in the next chapter. But in this chapter, as an encouragement to keep going, we'll continue our informal discussions, and show how to develop the tree methods we used in **PL** to prove that some **QL** inferences are valid.

25.1 The ∀-instantiation rule

We introduced **PL** trees as a fast-track, working-backwards, way of looking for a 'bad line' on a truth-table. However, we soon dropped the explicit talk of valuations, and used unsigned trees. And as we noted in Chapter 20, these unsigned trees (initially introduced as regimenting metalinguistic arguments) can also be thought of as reductio proofs couched in **PL**.

Here's the main idea again. We are given an inference

$A_1, A_2, \ldots, A_n \therefore C$

and want to know whether it is valid or not. Equivalently, we want to know whether the set of wffs

$A_1, A_2, \ldots, A_n, \neg C$

is or isn't inconsistent. We can show that this set *is* inconsistent (and so the original inference is valid) by constructing a closed tree with the wffs in question as its trunk. The basic principle which guides extending the tree is this: we extend an open branch *by adding to it further wffs which must also be true assuming that wffs already on the branch are true,* splitting the tree into two branches when we have to consider alternative cases (see §17.1).

We are now going to apply the very same idea to quantificational inferences. Except this time, we are going to *start* by thinking of trees the second way – i.e. not as metalinguistic arguments about valuations but as object-language arguments revealing that a bunch of wffs is inconsistent. (We have to do it this way round if we are going to introduce trees now, for we haven't yet met the appropriate official story about valuations for **QL** wffs. In what follows we rely on our grip on what **QL** wffs mean, and hence when they are true, given an interpretation of their basic vocabulary.)

Take our old friend, the argument ‘All philosophers are egocentric; Jack is a philosopher; hence Jack is egocentric’ (§1.2). Translated into **QL**, using the obvious code book, and taking the domain of quantification to be people, we have:

A $\mathsf{Fn}, \forall \mathsf{x}(\mathsf{Fx} \supset \mathsf{Gx}) \therefore \mathsf{Gn}$

(We’ll take it that the comma and the inference marker officially belong to **QL** as they do to **PL**.) We show this inference is valid by showing that the premisses and the negation of the conclusion form an inconsistent set. We start by setting down those wffs as the trunk of a tree:

(1) Fn
(2) $\forall \mathsf{x}(\mathsf{Fx} \supset \mathsf{Gx})$
(3) $\neg \mathsf{Gn}$

Now, how should we extend the tree? Assuming that those wffs *are* true together, what else has to be true? Well, if *everything* in some domain satisfies some condition, then any particular named thing in that domain satisfies that condition. So if something of the form $\forall v C(\ldots v \ldots v \ldots)$ is true, then $C(\ldots c \ldots c \ldots)$ will be true, given c names something in the domain. In this case, we are assuming ‘Fn’ is true, so ‘n’ certainly should name something in the domain (see §22.4). So the following ought also to be true:

(4) $(\mathsf{Fn} \supset \mathsf{Gn})$

And now we can just press on, applying the usual tree-building rule for wffs of the form $(A \supset B)$; and the tree closes.

(5) $\neg \mathsf{Fn}$ Gn
✻ ✻

The supposition that the premisses and negated conclusion of **A** are all true has been revealed to lead to contradiction. Which shows that the argument **A** is indeed valid.

Another simple example. Consider the inference

B $\forall \mathsf{x}(\mathsf{Fx} \supset \mathsf{Gx}), (\mathsf{Fn} \wedge \neg \mathsf{Hn}) \therefore \neg \forall \mathsf{x}(\mathsf{Gx} \supset \mathsf{Hx})$

which renders e.g. ‘All logicians are philosophers; Russell is a logician who isn’t wise; so not all philosophers are wise.’ Again, we assume that the premisses and negation of the conclusion are true, and set down those assumptions as the beginning of a tree.

(1) $\forall \mathsf{x}(\mathsf{Fx} \supset \mathsf{Gx})$
(2) $(\mathsf{Fn} \wedge \neg \mathsf{Hn})$
(3) $\neg\neg \forall \mathsf{x}(\mathsf{Gx} \supset \mathsf{Hx})$

Now we can apply the familiar rules for unpacking the connectives:

(4) Fn
(5) $\neg \mathsf{Hn}$
(6) $\forall \mathsf{x}(\mathsf{Gx} \supset \mathsf{Hx})$

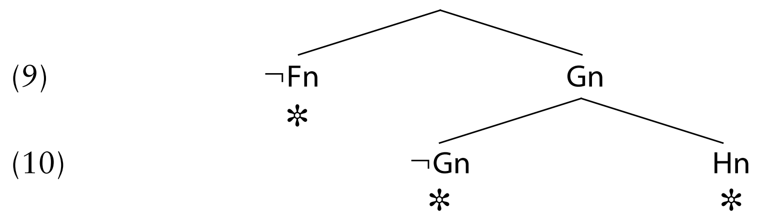
So far, so straightforward. To proceed further, we evidently need to extract some more 'n'-relevant information. (1) says that any thing *x* satisfies the condition '(Fx ⊃ Gx)'. So this will be true in particular of what 'n' denotes. Whence

(7) (Fn ⊃ Gn)

Similarly from (6) we have

(8) (Gn ⊃ Hn)

Now it is easy to extract an inconsistency. Using the standard rule for the material conditional twice, we have:



And we are done. The assumption that the premisses and the negated conclusion of **B** are all true together leads to contradictions however we try to work through the consequences of that assumption; so the argument is valid.

The new rule for tree-building which we are using here is the ∀-*instantiation rule* (we'll call it '(∀)' for short). The basic idea is that, given a universal quantification, then any substitution instance of it must also be true: i.e. any wff which is got by stripping off the quantifier and replacing the remaining occurrences of the associated variable with some name of an object in the domain must also be true. But any constant we are already using *is* supposed to name something in the domain; so that motivates the following schematic statement of the rule:

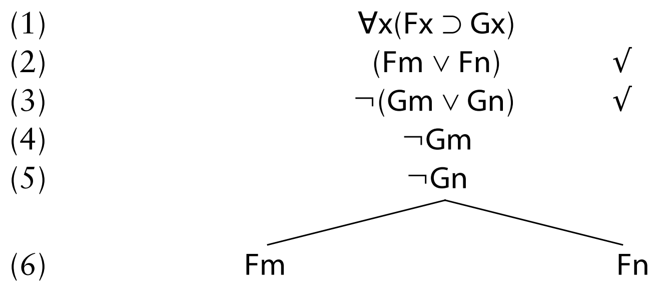
(∀) If ∀*vC*(…*v*…*v*…) appears on an open path, we can add *C*(…*c*…*c*…) to the foot of the path, where *c* is any constant which already appears on the path.

Note very carefully that – basically to keep our trees neatly pruned – this rule doesn't insist that we add *all* such instances of the quantified statement to *every* relevant path whenever we deploy it: *the rule simply permits us to draw out implications as and where needed.* And because we don't (and indeed can't) 'use up' all the content of the quantified wff in a single application of (∀), we *don't* check off the quantified wff after applying the instantiation rule.

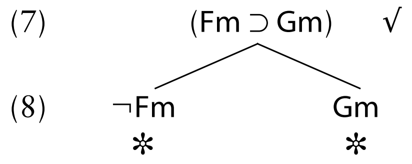
Here's a slightly more complex example of our new rule in operation, which illustrates the last point. Consider the argument 'Every great logician is also a great philosopher; either Moore or Russell is a great logician; so either Moore or Russell is a great philosopher.' Translating this into **QL**, and keeping the domain of quantification to be people, the inference is

C ∀x(Fx ⊃ Gx), (Fm ∨ Fn) ∴ (Gm ∨ Gn)

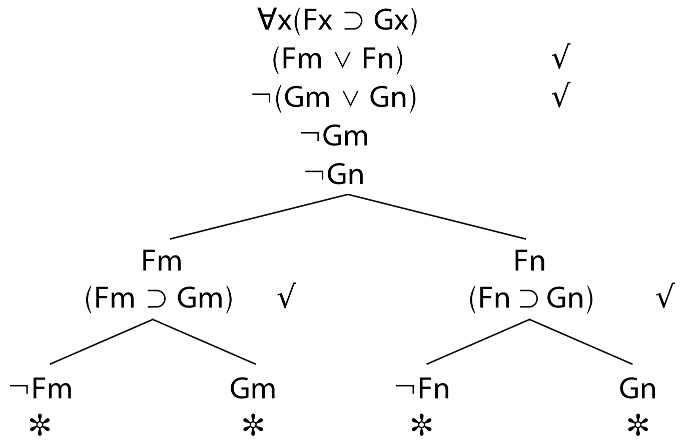
This is plainly valid. Still, let's *prove* that by supposing, in the usual way, that the premisses are true and the conclusion false, and aiming for contradictions.



We can now use (∀) to extract more ‘m’-related information, instantiate (1) with the name ‘m’ and continue the left-hand branch:



Next, consider the right-hand branch. We could have added the wff ‘(Fm ⊃ Gm)’ on this branch too, but that won’t do any useful work. What we need to do, in order to develop the right-hand branch, is extract ‘n’-related information from (1) using (∀) again. Now we see the importance of *not* having checked off (1), and marked it as ‘used up’. We hadn’t squeezed all the useful juice out of that quantified wff; we need to revisit it and extract more implications. This second application (∀) enables us to finish the tree.



The tree closes, and the inference **C** is shown to be valid.

25.2 Rules for negated quantifiers

We’ve stated the tree-building rule for deducing implications from a wff of the form ∀*vC*(…*v*…*v*…). What about a tree-building rule for dealing with a negated universal quantification, i.e. a wff of the form ¬∀*vC*(…*v*…*v*…)?

Well, ¬∀*vC*(…*v*…*v*…) is true just so long as ∃*v*¬*C*(…*v*…*v*…) is true (see §23.1). Hence a possible rule for treating negated universal quantifications is to transform them into equivalent existential quantifications, and pass them over to be dealt with by the rule (∃) for dealing with existential wffs – whatever that rule later turns out to be!

Similarly, note that ¬∃*vC*(...*v*...*v*...) is true just so long as ∀*v*¬*C*(...*v*...*v*...) is. Hence a possible rule for treating negated existential quantifications is to transform them into equivalent universal quantifications, and pass them back to be dealt with by the rule for (∀).

We'll adopt these rules for **QL** tree-building. Given a wff that *starts* with a negated quantifier, we can move the initial negation sign past the quantifier, swapping '∀' with '∃' as it passes. Since that transformation extracts all the content from the original wff – it takes us from one wff to an exactly equivalent wff – this time we *can* check it off after the rule is applied. So schematically:

(¬∀)	If ¬∀*vC*(...*v*...*v*...) appears on an open path, then we add the corresponding wff ∃*v*¬*C*(...*v*...*v*...) to each open path that contains it, and check it off.
(¬∃)	If ¬∃*vC*(...*v*...*v*...) appears on an open path, then we add the corresponding wff ∀*v*¬*C*(...*v*...*v*...) to each open path that contains it, and check it off.

Let's review two quick examples of these new rules at work. Consider

D ¬∃x(Fx ∧ Gx), Fm ∴ ¬Gm

(Perhaps that translates 'No logicians are wise; Russell is a logician; so Russell isn't wise.') The completed tree looks like this:

```
(1)               ¬∃x(Fx ∧ Gx)          √
(2)                    Fm
(3)                   ¬¬Gm
(4)               ∀x¬(Fx ∧ Gx)
(5)                ¬(Fm ∧ Gm)           √
                   /        \
(6)            ¬Fm            ¬Gm
                *              *
```

At (4) we have applied the new (¬∃) rule to (1); and then we applied (∀) at the next step, and the tree almost immediately closes.

Here's another intuitively valid argument:

E (Fm ∨ Fn), ∀x(Fx ⊃ Gx)∴ ∃xGx

(Compare: 'Either Jack or Jill is a student; all students are poor; hence there is someone who is poor.'). The tree starts in the obvious way

```
(1)                 (Fm ∨ Fn)           √
(2)                ∀x(Fx ⊃ Gx)
(3)                  ¬∃xGx              √
(4)                  ∀x¬Gx
                   /        \
(5)           Fm                Fn
```

We now proceed as on the tree for **C**. First we want to instantiate the two universal quantifications (2) and (4) with ‘m’ to get ‘m’-related information useful for extending the left-hand branch. But there would be no point in adding the resulting wffs to the right-hand branch too (though it wouldn’t be *wrong* to do so, just redundant). Rather, we need to extract ‘n’-related information useful for the right-hand branch. So we continue:

```
                |                          |
(6)        (Fm ⊃ Gm)                  (Fn ⊃ Gn)      (from 2)
(7)           ¬Gm                        ¬Gn         (from 4)
```

Then applying the rule for the conditional twice will yield:

```
                 /\                        /\
(8)        ¬Fm        Gm             ¬Fn        Gn
            *          *              *          *
```

25.3 The ∃-instantiation rule

Consider next the following argument

F ∃xFx, ∀x(Fx ⊃ Gx) ∴ ∃xGx

(Perhaps: ‘Someone is a great logician; all great logicians are cool; hence someone is cool’). Assume that the premisses and the negation of the conclusion are all true, and then apply the rule (¬∃), and we get

```
(1)                 ∃xFx
(2)              ∀x(Fx ⊃ Gx)
(3)                ¬∃xGx            √
(4)                ∀x¬Gx
```

where we have applied the (¬∃) rule to get to line (4).

(1) tells us that at least one thing in the domain satisfies ‘F’. Now, there can’t be any harm in picking out one of these things and arbitrarily *dubbing* it. If the original wffs including (1) are consistent, no contradiction can ensue just from this act of selecting and naming. Let’s use a constant that isn’t currently being used for other purposes – e.g. ‘a’ (for ‘arbitrary’!). So by hypothesis we have

```
(5)                  Fa
```

What else do we know about the thing now dubbed ‘a’? Given the universal claims (2) and (4), we can deduce particular claims about it as follows:

```
(6)               (Fa ⊃ Ga)          √
(7)                  ¬Ga
                     /\
(8)          ¬Fa             Ga
              *               *
```

The tree closes. To repeat, the mere act of dubbing one of the things which are ‘F’ can’t have created the contradiction. So the contradiction must have been in the original assignments of truth to each of (1) to (3). Hence there can be no way of making (1) to (3) all true, and so the inference **F** is valid.

Another quick example. Take the evidently valid syllogism ‘All logicians are philosophers; all philosophers are wise; so all logicians are wise.’ Using the obvious translation we get,

G $\forall x(Fx \supset Gx), \forall x(Gx \supset Hx) \therefore \forall x(Fx \supset Hx)$

We start the tree in the standard way

(1) $\forall x(Fx \supset Gx)$
(2) $\forall x(Gx \supset Hx)$
(3) $\neg\forall x(Fx \supset Hx)$

and apply the ($\neg\forall$) rule to get

(4) $\exists x\neg(Fx \supset Hx)$

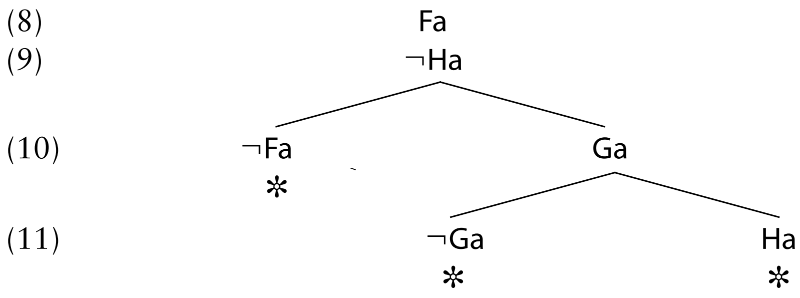
(4) tells us that there is at least one thing in the domain such that the condition expressed by ‘$\neg(Fx \supset Hx)$’ holds of it. Pick one, and let’s dub it with a name not otherwise being used in this argument, say ‘a’ again. Then we have

(5) $\neg(Fa \supset Ha)$

Applying the ($\forall$) rule to (1) and (2), we get the following further information about this thing dubbed ‘a’:

(6) $(Fa \supset Ga)$
(7) $(Ga \supset Ha)$

Then applying the standard propositional rules to (5) to (7) in turn yields



So the tree closes. Again, the contradictions here cannot have been created by the mere act of selecting and then dubbing something with a name not otherwise in use. Therefore the trouble must be in the original supposition that (1) to (3) are all true; so argument **G** is valid.

Note that absolutely crucial qualification: contradictions cannot be created by the mere act of dubbing something *with a name not otherwise in use*. Of course, if a name has already been recruited for some other purpose, then using it again to dub something may very well cause trouble, as we shall now see.

Consider the following very simple case. The inference ‘Someone is a great philosopher: hence Derrida is a great philosopher’ is obviously bogus. Similarly the inference

H $\exists xFx \therefore Fn$

is equally invalid. Now suppose we start a tree, and then – at line (3) – instantiate the existential premiss with the name already in use, to get

(1) ∃xFx
(2) ¬Fn
(3) Fn
✻

We have then absurdly 'proved' that the fallacious argument **H** is valid!

Here's another example of the same mistake getting us into trouble. Consider the inference:

I ∃xFx, ∃xGx ∴ ∃x(Fx ∧ Gx)

Suppose we set out to build a quantifier tree, starting:

(1) ∃xFx
(2) ∃xGx
(3) ¬∃x(Fx ∧ Gx)

We apply the (¬∃) rule to get

(4) ∀x¬(Fx ∧ Gx)

Now, if there is at least one thing which satisfies 'F', it can do no harm to pick one and dub it 'a', so we could write

(5) Fa

Likewise, if there is at least one thing which satisfies 'G', it can do no harm to pick one and dub it with some name. But suppose we offend by wrongly using the same name again, and add

(6) Ga

to the tree. Given (4), we could continue

(7) ¬(Fa ∧ Ga) √

(8) ¬Fa ¬Ga
✻ ✻

The tree then closes, giving the verdict that argument **I** is valid, which it isn't! Compare 'Someone is very tall; someone is very short; hence someone is very tall and very short.'

It's evidently step (6) that caused the trouble. To repeat, although (2) tells us that something satisfies 'G', and it can do no harm to pick out such a thing and dub it, *we mustn't suppose that it is the same thing that we have already dubbed 'a'*. At that step (6), we are only entitled to add something like

(6) Gb

We could continue

(7) ¬(Fa ∧ Ga)

as before, and also add

(8)	¬(Fb ∧ Gb)	

But this plainly isn't going to get the tree to close.

In summary, then, all this motivates the following ∃-*instantiation rule*:

(∃) Given an existential quantification of the form ∃*v*C(…*v*…*v*…) on an open path, then we can add C(…*c*…*c*…) to each open path that it is on, where *c* is a 'new' constant, i.e. one which *hasn't yet appeared on any of those paths*; we then check off the original wff.

∃*v*C(…*v*…) tells us that something – and it may be only one thing – is *C*. So picking such a thing (and dubbing it) exhausts the claim's commitments: that's why we need to check it off once the rule (∃) is applied.

Our two proposed rules for un-negated quantifiers might look confusingly similar, so let's point up the essential difference loud and clear.

- The rule (∀) says that you can instantiate a universal quantifier with any name *already in use (on the path).*
- The (∃) rule says that you can instantiate an existential quantifier with any name *that isn't already in use (on the relevant paths).*

In Chapter 28, we'll return to motivate these quantifier tree-building rules much more carefully (but we'll need a proper account of the semantics for **QL** first). However, our quick-and-dirty explanations will suffice for the moment.

25.4 More examples

To help fix ideas, let's show that the following simple arguments are all valid by translating into **QL** and using trees (try these examples before reading on):

J Some chaotic attractors are not fractals; all Cantor sets are fractals; hence some chaotic attractors are not Cantor sets.

K If there is a logician who isn't a philosopher, then Jones isn't a philosopher. Jones *is* a philosopher. So every logician is a philosopher.

L Only if Owen is happy does he love someone. If anyone loves Nerys then Owen does. Maldwyn loves Nerys. So Owen is happy.

The first should raise no translational difficulties:

J′ ∃x(Fx ∧ ¬Gx), ∀x(Hx ⊃ Gx) ∴ ∃x(Fx ∧ ¬Hx)

and the finished tree looks like this (still adding line numbers for ease of reference):

(1)	∃x(Fx ∧ ¬Gx)	√
(2)	∀x(Hx ⊃ Gx)	
(3)	¬∃x(Fx ∧ ¬Hx)	√
(4)	∀x¬(Fx ∧ ¬Hx)	
(5)	(Fa ∧ ¬Ga)	√
(6)	(Ha ⊃ Ga)	√
(7)	¬(Fa ∧ ¬Ha)	√

```
(8)                          Fa
(9)                         ¬Ga
                    ┌────────┴────────┐
(10)               ¬Ha                Ga
                    │                 ✻
           ┌────────┴────────┐
(11)      ¬Fa              ¬¬Ha
           ✻                 ✻
```

How was this tree produced? The first three lines are automatic, and applying the (¬∃) rule immediately gives (4). That yields an existentially quantified wff and two universally quantified wffs unchecked.

Now, a useful rule of thumb is *where possible apply the* (∃) *rule before the* (∀) *rule*. Why? Because instantiating an existential quantification always introduces a *new* name, and you'll probably need to go back to the universal quantifications *afterwards* to extract any needed information about the named object. But in fact, in this case, we don't get a choice anyway. The only rule we can apply once we've got to line (4) is the (∃) rule to (1). So we pick one of the things which makes '∃x(Fx ∧ ¬Gx)' true, and dub it 'a', yielding (5), and checking off (1). We then extract 'a'-relevant information from (2) and (4) by use of the (∀) rule. With (6) and (7) in place, the rest is straightforward.

We can translate the next argument thus:

K′ (∃x(Fx ∧ ¬Gx) ⊃ ¬Gn), Gn ∴ ∀x(Fx ⊃ Gx)

Note carefully the bracketing in the translation of the first premiss. The tree then starts in the standard way, applying the rule (¬∀) to (3):

```
(1)                 (∃x(Fx ∧ ¬Gx) ⊃ ¬Gn)          √
(2)                          Gn
(3)                      ¬∀x(Fx ⊃ Gx)              √
(4)                      ∃x¬(Fx ⊃ Gx)
                  ┌───────────┴───────────┐
(5)         ¬∃x(Fx ∧ ¬Gx)                ¬Gn
                  |                        ✻
```

The left-hand branch now continues

```
(6)         ∀x¬(Fx ∧ ¬Gx)
(7)            ¬(Fa ⊃ Ga)
```

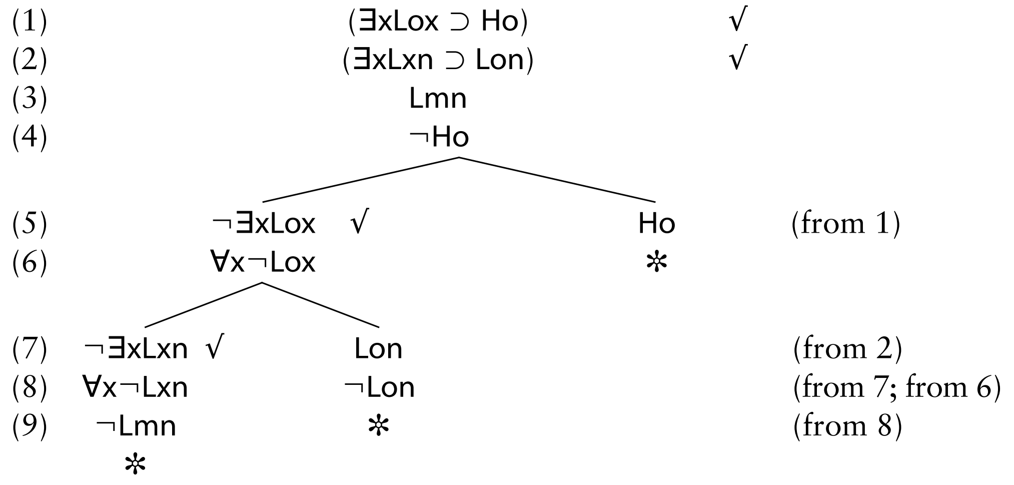
applying first the (¬∃) rule to (5), and then the (∃) rule to (4). Then, after applying the (∀) rule to (6), we quickly finish:

```
(8)          ¬(Fa ∧ ¬Ga)
(9)               Fa
(10)             ¬Ga
          ┌───────┴───────┐
(11)     ¬Fa            ¬¬Ga
(12)      ✻               ✻
```

Finally, here's a translation of the third argument (why the *existential* quantifiers for 'anyone'? – compare §24.1, Example 10):

L′ (∃xLox ⊃ Ho), (∃xLxn ⊃ Lon), Lmn ∴ Ho

The tree is then fairly straightforward:



Note that on the right-hand branch at (8), we could have instantiated the universal quantifier at (6) by any of the names 'm', 'n', or 'o': our choice, however, is the sensible one, as it makes the right-hand branch close. Similarly at (9), we make the obvious choice of how best to instantiate the quantifier at (8).

25.5 Open and closed trees

We've introduced the language **QL**. We've explained four rules that we need in addition to the rules for connectives in order to construct **QL** tree arguments to show various inferences are valid. There's a little work to be done tidying up the syntax for our formal language; and there's rather a lot of work to be done developing the semantics. We need to explore trees further, and to discuss whether any more tree-building rules are necessary to give us a 'complete' system (compare §20.4). In fact, we'll see that we do need to slightly augment the rules we've so far stated. But that is all to come later. For the moment, let's concentrate on emphasizing a crucial point which should be made straight away.

Predicate logic really comes into its own when we are dealing with inferences involving propositions that involve more than one quantifier. We'll look at a range of examples in later chapters; but here's just one to be going on with (and to set the scene for the point we need to make). Consider the argument 'Everyone loves someone who is kind; no cruel person is kind; Caligula is cruel. Still, it follows that not everyone is cruel'. That's evidently valid, if we read the first premiss as meaning everyone loves at least one kind person. For take Caligula. Although he is cruel, he loves someone who is kind. And that person, being kind, isn't cruel. So not everyone is cruel.

Let's warrant this argument with a tree proof. Here's a translation:

M ∀x∃y(Fy ∧ Lxy), ∀x(Gx ⊃ ¬Fx), Gn ∴ ¬∀xGx

The tree starts in the obvious way.

$$\begin{array}{lcl}
(1) & \forall x\exists y(Fy \wedge Lxy) & \\
(2) & \forall x(Gx \supset \neg Fx) & \\
(3) & Gn & \\
(4) & \neg\neg\forall xGx & \surd \\
(5) & \forall xGx &
\end{array}$$

We now have three universal quantifiers to instantiate, using the rule (∀). But we'll tackle (1) first, so we can 'get at' the embedded existential quantifier. At the moment, we only have one constant in play, so we have to instantiate (1) with that, to yield

$$\begin{array}{lc}
(6) & \exists y(Fy \wedge Lny)
\end{array}$$

Now we'll follow the heuristic rule we noted before: where possible, we apply the (∃) rule before the (∀) rule. We have to instantiate (6) with a name *new* to the argument:

$$\begin{array}{lcl}
(7) & (Fa \wedge Lna) & \surd \\
(8) & Fa & \\
(9) & Lna &
\end{array}$$

We can now continue by extracting more 'a'-relevant information by instantiating the universal quantifications (2) and (5):

$$\begin{array}{lcccl}
(10) & & (Ga \supset \neg Fa) & & \surd \\
(11) & & Ga & & \\
 & \swarrow & & \searrow & \\
(12) & \neg Ga & & \neg Fa & \\
 & * & & * &
\end{array}$$

And once more we are done.

So far, so good. But note we could have taken a wrong turning after line (7). For we might be struck by the fact that we now have the new name 'a' in play, and we could instantiate (1) again with this new name, thus …

$$\begin{array}{lc}
(8') & \exists y(Fy \wedge Lay)
\end{array}$$

We then have a new existential quantification which we can instantiate in turn with a constant new to the argument:

$$\begin{array}{lc}
(9') & (Fb \wedge Lab)
\end{array}$$

Now suppose we (unwisely, but quite legitimately) use the name 'b' to instantiate (1) again to infer

$$\begin{array}{lc}
(10') & \exists y(Fy \wedge Lby)
\end{array}$$

and instantiate *this* with respect to a new name:

$$\begin{array}{lc}
(11') & (Fc \wedge Lbc)
\end{array}$$

Evidently we could go on, and on, and on without end. The unwise strategy of returning to (1) whenever we can instantiate it with a new name could prevent the tree from ever closing! (Query: but what entitles us to suppose that there are all those different things in the domain to be named 'a', 'b', 'c', ...? Maybe the domain only contains one thing! Reply: the question is based on a misunderstanding. We are introducing a sequence of new *names* one after another; but maybe the names refer to the same *thing*, maybe to different things: that is left open.)

So here we encounter a very deep and important difference between **PL** and **QL** trees. In the propositional case it doesn't matter (except for the aesthetics) in which *order* the available rules are applied. If a tree closes, then any legally constructed tree with the same initial trunk will also close (see §§16.4, 19.3). *In the quantificational case, by contrast, the order in which rules are applied matters.* Our first tree for **M** closed: but we've seen that a second tree, equally constructed in accordance with the rules, could be extended without limit, without closing.

Now, finding *one* well-constructed closed tree for a **QL** argument is enough to prove that the argument is valid. If *one* line of reasoning from the assumption that the premisses and the negation of the conclusion are true leads to contradictions all round, that settles it: if the premisses are true, the conclusion must be true too. This result isn't subverted by the fact that some *other* line of reasoning trails off in the wrong direction and fails to reveal a contradiction.

How can **QL** trees have branches going on for ever when that can't happen for **PL** trees? Because our rules for unpacking complex **PL** wffs systematically 'use up' wffs, always taking us to shorter and shorter wffs: and – assuming there are only a finite number of premisses to begin with – eventually we will simply run out of rules to apply. We must end up with a completed tree, either open or closed. In the quantificational case, however, we can get into cycles like the one that takes us from (6) to (8′) to (10′) to ..., where the wffs don't become any simpler, and so the tree never finishes.

Of course, our second, un-ending, tree for **M** is constructed according to a pretty crazy strategy. And in fact, there is a decently systematic strategy which will always deliver us a closed tree for an argument *if there is one*. But intriguingly, there is no mechanical test which will show us (in finite time) whether applying this strategy will eventually deliver a closed tree; there is no mechanical test for whether a quantified argument is valid or not. More on this anon (see Chapter 30).

But having seen where all this development of **QL** is taking us, and in particular having had a glimpse of how we can develop a tree method for warranting inferences, we really need now to return to basics, to tidy up the details. That's the business for the next few chapters.

25.6 Summary

- We have introduced four new rules for quantifier tree-building, which we've labelled (∀), (¬∀), (∃), (¬∃), and given some motivation for them. We'll state the rules for **QL** trees more carefully in Chapter 29.

- **QL** trees give us *sound* proofs of validity: i.e. when a properly constructed tree closes, the argument whose premisses and negated conclusion start the tree is indeed valid. But we noted that valid arguments can also generate (infinite) open trees.
- **QL** trees are not a *mechanical method* for demonstrating the validity or invalidity of arguments. There can't be one.

Exercises 25

A Show the following simple arguments are valid by translating into **QL** and using trees.

1. Everyone is rational; hence Socrates is rational.
2. No-one loves Angharad; hence Caradoc doesn't love Angharad.
3. No philosopher speaks Welsh; Jones is a philosopher; hence Jones does not speak Welsh.
4. Jones doesn't speak Welsh; hence not everyone speaks Welsh.
5. Socrates is rational; hence someone is rational.
6. Some philosophers speak Welsh; all Welsh speakers sing well; hence some philosophers sing well.
7. All electrons are leptons; all leptons have half-integral spin; hence all electrons have half-integral spin.
8. All logicians are philosophers; all philosophers are rational people; no rational person is a flat-earther; hence no logician is a flat-earther.
9. If Jones is a bad philosopher, then some Welsh speaker is irrational; but every Welsh speaker is rational; hence Jones is not a bad philosopher.

B Consider the following rule

($\neg\exists'$) If $\neg\exists vC(\ldots v \ldots v \ldots)$ appears on an open path, then we can add $\neg C(\ldots c \ldots c \ldots)$ to that path, where c is any constant which already appears on the path.

Show informally that this rule would do as well as our rule ($\neg\exists$). What would be the analogous rule for dealing with negated universal quantifiers without turning them first into existential quantifiers?

C Suppose we had set up predicate logic with a *single* quantifier formed using the symbol 'N', so that $\mathsf{N}vCv$ holds when nothing is C. Show that the resulting language would be expressively equivalent to our now familiar two-quantifier language **QL**. What would be an appropriate set of rules for tree-building in a language with this single quantifier?

29

More on QL trees

This chapter begins by gathering together the rules for **QL** trees introduced in Chapter 25. We look at some more complex examples using closed trees to demonstrate q-validity. We then explain a needed small addition to one of the rules. Finally, we consider what can be learnt from looking at trees that *don't* close.

29.1 The official rules

In the boxes over the next two pages, we summarize the rules for building the unsigned **QL** trees that we've already encountered. Rules (a) to (i) are the familiar rules for the connectives, trivially re-ordered. Then we add the current four rules for quantifiers. How are these rules to be interpreted and applied?

First, a quick reminder of what we mean when we say that a wff *W* is e.g. of the form $(A \supset B)$ or $\forall v C(\ldots v \ldots v \ldots)$. We mean, in the first case, that *W* can be constructed from the template '$(A \supset B)$' by replacing '*A*' and '*B*' by **QL** wffs: in that case, '$\supset$' is the main logical operator of *W* (see §26.4). In the second case, we mean that *W* can be constructed by prefixing the universal quantifier $\forall v$ to a wff where the variable *v* has only *free*, i.e. unbound, occurrences: the quantifier $\forall v$ is the main logical operator of *W*.

In each case, then, the tree-building rules tell us how to extend a tree depending on the *main* operator of the wff *W* which is being processed. Given the wff

$$(\forall x(Fx \land Gx) \supset Fn)$$

we can apply the appropriate tree-splitting rule (g), to deal with '$\supset$'; but we *cannot* apply the rule ($\forall$), as '$\forall x$' is not the main operator. Likewise, given the wff

$$\exists x \neg\neg(Fx \land Gx)$$

we can apply the quantifier rule ($\exists$) but not the double-negation rule (a). And so on.

Given a **QL** inference (involving only closed wffs, as we said before) we build a corresponding tree in the following stages, just as with a **PL** inference:

(1) Start off the trunk of the tree with the premisses A_1, A_2, …, A_n and the negation of conclusion *C* of the inference under consideration.

(2) Inspect any not-yet-closed path down from the top of the tree to see whether it involves an explicit contradiction, i.e. for some wff *W* it

*Rules for building **QL** trees*

Suppose that W is a non-primitive wff on an open branch. We can then apply the corresponding rule:

(a) W is of the form $\neg\neg A$: add A to each open path containing W. Check off W. Schematically,

$$\begin{tikzcd} \neg\neg A \quad \surd \arrow[d, dash] \\ A \end{tikzcd}$$

(b) W is of the form $(A \land B)$: add both A and B to each open path containing W. Check off W.

(c) W is of the form $\neg(A \lor B)$: add $\neg A$ and $\neg B$ to each open path containing W. Check off W.

(d) W is of the form $\neg(A \supset B)$: add A and $\neg B$ to each open path containing W. Check off W.

$$\begin{tikzcd} (A \land B)\ \surd \arrow[d, dash] \\ A \\ B \end{tikzcd} \qquad \begin{tikzcd} \neg(A \lor B)\ \surd \arrow[d, dash] \\ \neg A \\ \neg B \end{tikzcd} \qquad \begin{tikzcd} \neg(A \supset B)\ \surd \arrow[d, dash] \\ A \\ \neg B \end{tikzcd}$$

(e) W is of the form $(A \lor B)$: add a new fork to each open path containing W, with branches leading to the alternatives A and B. Check off W.

(f) W is of the form $\neg(A \land B)$: add a new fork to each open path containing W, with branches leading to the alternatives $\neg A$ and $\neg B$. Check off W.

(g) W is of the form $(A \supset B)$: add a new fork to each open path containing W, with branches leading to the alternatives $\neg A$ and B. Check off W.

(h) W is of the form $(A \equiv B)$: add a new fork to each open path containing W, leading to the alternatives A, B and $\neg A$, $\neg B$. Check off W.

(i) W is of the form $\neg(A \equiv B)$: add a new fork to each open path containing W, leading to the alternatives A, $\neg B$ and $\neg A$, B. Check off W.

$$\begin{tikzcd} & (A \lor B)\ \surd \arrow[dl, dash] \arrow[dr, dash] & \\ A & & B \end{tikzcd} \qquad \begin{tikzcd} & \neg(A \land B)\ \surd \arrow[dl, dash] \arrow[dr, dash] & \\ \neg A & & \neg B \end{tikzcd} \qquad \begin{tikzcd} & (A \supset B)\ \surd \arrow[dl, dash] \arrow[dr, dash] & \\ \neg A & & B \end{tikzcd}$$

$$\begin{tikzcd} & (A \equiv B)\ \surd \arrow[dl, dash] \arrow[dr, dash] & \\ A & & \neg A \\ B & & \neg B \end{tikzcd} \qquad \begin{tikzcd} & \neg(A \equiv B)\ \surd \arrow[dl, dash] \arrow[dr, dash] & \\ A & & \neg A \\ \neg B & & B \end{tikzcd}$$

Rules for building **QL** *trees, continued*[*]

(¬∀) *W* is of the form $\neg\forall vC$: add $\exists v\neg C$ to each open path on which *W* appears. Check off *W*.

(¬∃) *W* is of the form $\neg\exists vC$: add $\forall v\neg C$ to each open path on which *W* appears. Check off *W*.

$$\begin{array}{c} \neg\forall vC \quad \surd \\ | \\ \exists v\neg C \end{array} \qquad\qquad \begin{array}{c} \neg\exists vC \quad \surd \\ | \\ \forall v\neg C \end{array}$$

(∀) *W* is of the form $\forall vC(\ldots v\ldots v\ldots)$: add $C(\ldots c\ldots c\ldots)$ to an open path on which *W* appears, where *c* is a constant on that path (which hasn't already been used to instantiate *W* on that path). *W* is *not* checked off.

$$\begin{array}{c} \forall vC(\ldots v\ldots v\ldots) \\ | \\ C(\ldots c\ldots c\ldots) \quad [c \text{ old}] \end{array}$$

(∃) *W* is of the form $\exists vC(\ldots v\ldots v\ldots)$: add $C(\ldots c\ldots c\ldots)$ to each open path on which *W* appears, where *c* is a constant *new* to the paths. Check off *W*.

$$\begin{array}{c} \exists vC(\ldots v\ldots v\ldots) \quad \surd \\ | \\ C(\ldots c\ldots c\ldots) \quad [c \text{ new}] \end{array}$$

contains both *W* and also the corresponding formula ¬*W*. If it does, then we *close* that path with a star as an absurdity marker.

(3) If we have run out of rules to apply to unchecked non-primitive formulae left on any *open* path, then we have to stop!

(4) Otherwise, we choose some unchecked non-primitive formula *W* that sits on an *open* path. We then apply the appropriate rule, and extend the tree by adding more (closed) wffs in the described way. We can then check off *W*, except after an application of rule (∀).

(5) Loop back to step (2).

As we saw (§25.5), this process may not terminate in the case of **QL** trees – and it is possible for injudicious choices at step (4) of the tree-building process to lead to a non-terminating tree even though *other* choices *would* lead to a closed tree. Why do we check off a wff *W* when we apply the appropriate tree-building rule, *except* when *W* is universally a quantified wff and we are applying the (∀) rule? Because in the other cases, the unpacking rules effectively exhaust the content of the wff we are unpacking (and there is no point in doing that twice over).

[*] But NB the extended version of the (∀) rule in §29.3!

By contrast, a universal quantified wff tells us that *everything* in the domain satisfies a certain condition. We can of course infer that a particular named object in the domain satisfies that condition. But there is more content to the original quantified claim than is given by that particular implication of it.

And why does the (∀) rule tell us not to re-instantiate a universally quantified wff using the same constant again on the same path? Because *that* would be quite redundant. It would just put a duplicate of the same wff on the same path.

29.2 Further examples of closed trees

Let's consider more inferences involving wffs with multiple quantifiers. Take first:

A Some people are boastful. No one likes anyone boastful. So some people aren't liked by anyone.

To confirm the argument is valid, translate it into **QL**, e.g. by

$\exists xFx, \forall x\forall y(Fy \supset \neg Lxy) \therefore \exists x\forall y\neg Lyx$

where the translation manual we are using is plain. Then the tree starts:

(1) $\exists xFx$
(2) $\forall x\forall y(Fy \supset \neg Lxy)$
(3) $\neg\exists x\forall y\neg Lyx$

Before we continue, let's set down a couple of handy rules of thumb for applying quantifier tree-building rules in a sensible order:

- At each step, deal with any negated quantifiers first.
- Instantiate existential quantifiers before universals (see §25.4).

Applying these in turn, the tree will continue …

(4) $\forall x\neg\forall y\neg Lyx$

Note, by the way, that the rule (¬∃) applied to (3) only pushes the negation sign past the *initial* quantifier. Next, instantiating (1) with a new constant, we have

(5) Fa

Now how shall we proceed? We eventually want to instantiate the *inner* universal quantifier of (2), to get another occurrence of 'Fa'; but rule (e) does not allow us to instantiate inner quantifiers; the tree-building rules insist we proceed step by step, from the outside in. So let's instantiate (4) instead:

(6) $\neg\forall y\neg Lya$

We've now got another negated quantifier, so deal with that next …

(7) $\exists y\neg\neg Lya$

This gives us a new existential quantification, so the second rule of thumb enjoins us now to deal with *that*, instantiating (7) with a new name, of course:

(8) $\neg\neg Lba$

Now return to instantiate the initial quantifier in (2) with 'b'; that yields another universally quantified wff, we can instantiate by 'a':

(9) $\forall y(Fy \supset \neg Lby)$
(10) $(Fa \supset \neg Lba)$

$$\begin{tikzcd}[column sep=small] & (Fa \supset \neg Lba) \arrow[dl, dash] \arrow[dr, dash] & \\ \neg Fa & & \neg Lba \end{tikzcd}$$

(11) $\neg Fa$ ✼ $\quad \neg Lba$ ✼

And we are done. The inference is q-valid, and hence valid. (Make sure you see why it was sensible to instantiate (2) with ‘b’ rather than ‘a’.)

Consider next

B Everyone admires some logician; whoever admires someone respects them. So everyone respects some logician.

The translation of the premisses and negated conclusion shouldn’t cause trouble by now (just note that the second premiss means that for *any* pair, if the first admires the second, then the first respects the second):

(1) $\forall x\exists y(Fy \wedge Lxy)$
(2) $\forall x\forall y(Lxy \supset Mxy)$
(3) $\neg\forall x\exists y(Fy \wedge Mxy)$

Checking off (3), the next steps are more or less automatic.

(4) $\exists x\neg\exists y(Fy \wedge Mxy)$ √
(5) $\neg\exists y(Fy \wedge May)$ √
(6) $\forall y\neg(Fy \wedge May)$

We *could* instantiate (6) using ‘a’, but that would be pointless. So let’s next instantiate (1) and (2) instead:

(7) $\exists y(Fy \wedge Lay)$
(8) $\forall y(Lay \supset May)$

Next, instantiating the existential quantification first (following our rule of thumb), and remembering to use a new constant, we get

(9) $(Fb \wedge Lab)$

Using this new constant to instantiate the two remaining universal quantifiers at (6) and (8), we get

(10) $\neg(Fb \wedge Mab)$
(11) $(Lab \supset Mab)$

Finally the tree closes using the familiar connective rules:

(12) Fb
(13) Lab

$$\begin{tikzcd}[column sep=small] & Lab \arrow[dl, dash] \arrow[dr, dash] & & \\ \neg Fb & & \neg Mab \arrow[dl, dash] \arrow[dr, dash] & \\ & \neg Lab & & Mab \end{tikzcd}$$

(14) $\neg Fb$ ✼ $\quad \neg Mab$
(15) $\neg Lab$ ✼ $\quad Mab$ ✼

Let's next revisit an example from §5.4:

C No girl loves any sexist pig. Caroline is a girl who loves whoever loves her; Henry loves Caroline; hence Henry is not a sexist pig.

Rendered into **QL** using an obvious translation dictionary, the tree will start:

$$\begin{array}{ll}
(1) & \forall x(Gx \supset \neg\exists y(Fy \wedge Lxy)) \\
(2) & (Gm \wedge \forall x(Lxm \supset Lmx)) \\
(3) & Lnm \\
(4) & \neg\neg Fn
\end{array}$$

And the next few steps are straightforward. Just unpack the conjunction in (2), checking it off, and then instantiate (1) with respect to 'm':

$$\begin{array}{llll}
(5) & & Gm & \\
(6) & & \forall x(Lxm \supset Lmx) & \\
(7) & & (Gm \supset \neg\exists y(Fy \wedge Lmy)) \ \surd & \\
(8) & \neg Gm & & \neg\exists y(Fy \wedge Lmy) \ \surd \\
(9) & * & & \forall y\neg(Fy \wedge Lmy)
\end{array}$$

That leaves us with two universal quantifications, at (6) and (9), to instantiate. Instantiating (9) by 'n' will give us 'Fn' in the scope of a negation, which should contradict the earlier '¬¬Fn'. And it does …

$$\begin{array}{llll}
(10) & & \neg(Fn \wedge Lmn) \ \surd & \\
(11) & \neg Fn & & \neg Lmn \\
 & * & &
\end{array}$$

The obvious thing to do now is to instantiate (6) with respect to 'n' as well, to conclude the tree

$$\begin{array}{llll}
(12) & & (Lnm \supset Lmn) \ \surd & \\
(13) & \neg Lnm & & Lmn \\
 & * & & *
\end{array}$$

We are done: the inference is valid.

Our fourth sample argument is of a type which was recognized by medieval logicians as being valid, but couldn't be shown to be so within their logic of general propositions:

D All horses are mammals; so any horse's head is a mammal's head.

Modern logicians can do better than their predecessors here. In fact, as soon as we see how to translate the conclusion into **QL**, finding a tree to show that **D** is indeed q-valid is straightforward.

First, then, the translational problem. 'Any horse's head is a mammal's head' is equivalent to 'For any *x*, if *x* is the head of some horse, then *x* is the head of some mammal', i.e. 'for any *x*, if (for some horse *y*, *x* is the head of *y*), then (for some mammal *z, x* is the head of *z*)'. So, let's fix on the dictionary

'F' means … *is a horse*
'G' means … *is a mammal*
'M' means … *is the head of* …

and set the domain to be e.g. earthly things. Then **C**'s conclusion can be rendered

$$\forall x\{\exists y(Fy \wedge Mxy) \supset \exists z(Gz \wedge Mxz)\}$$

The tree will therefore commence

(1) $\forall x(Fx \supset Gx)$
(2) $\neg\forall x\{\exists y(Fy \wedge Mxy) \supset \exists z(Gz \wedge Mxz)\}$ √
(3) $\exists x\neg\{\exists y(Fy \wedge Mxy) \supset \exists z(Gz \wedge Mxz)\}$ √
(4) $\neg\{\exists y(Fy \wedge May) \supset \exists z(Gz \wedge Maz)\}$ √
(5) $\exists y(Fy \wedge May)$
(6) $\neg\exists z(Gz \wedge Maz)$

That's pretty much automatic given our two rules of thumb for tree-building. We dealt first with the negated quantifier at (2) to give (3). That leaves us with the universally quantified (1) and the existentially quantified (3), so we apply the 'existentials before universals' rule. Stripping off the quantifier '∃x', and replacing every occurrence of 'x' with a new name gives us (4). Note the bracketing in (4); it is a negated conditional. So applying the relevant rule gives us (5) and (6). Again, we first deal with the negated quantifier, and check off (6) to yield

(7) $\forall z\neg(Gz \wedge Maz)$

Now we have two universally quantified propositions at (1) and (7), and the existential quantification at (5). So next instantiate (5) with a new name:

(8) $(Fb \wedge Mab)$ √
(9) Fb
(10) Mab

and then the rest is straightforward. Instantiating (1) and (7) with respect to 'b' (why try 'b' rather than 'a'?), and then unpacking the connectives, we finish …

(11) $(Fb \supset Gb)$ √
(12) $\neg(Gb \wedge Mab)$ √

(13) $\neg Fb$ ✻ | Gb

(14) under Gb: $\neg Gb$ ✻ | $\neg Mab$ ✻

Finally in this section, we'll return to §5.1, and again consider

E Everyone loves a lover; Romeo loves Juliet; so everyone loves Juliet.

We saw that this is valid if the first premiss is interpreted to mean 'everyone loves anyone who is a lover' (i.e. 'everyone *x* is such that, take anyone *y*, if *y* loves someone then *x* loves *y*'). Using 'L' for *loves*, etc., we begin the tree

```
(1)              ∀x∀y(∃z Lyz ⊃ Lxy)
(2)                    Lmn
(3)                  ¬∀x Lxn
```

Following the rule of thumb ‘negated quantifiers first’, we can check off (3) and add:

```
(4)                  ∃x¬Lxn
```

Remembering the other tip, ‘existentials before universals’, we instantiate (4):

```
(5)                   ¬Lan
```

But now what? Let’s try instantiating the two quantifiers in (1) – one at a time! – with ‘a’ and ‘n’, which will make the consequent of the conditional ‘Lan’, preparing the way for a contradiction with (5). So we get

```
(6)               ∀y(∃z Lyz ⊃ Lay)
(7)                (∃z Lnz ⊃ Lan)      √
                   /            \
(8)           ¬∃z Lnz   √         Lan
(9)           ∀z¬Lnz               *
```

How do we carry on from here? Consider the argument we gave to show that this inference is valid in §5.1. We in fact had to use the first premiss twice there, at lines (4) and (7) of the informal proof. We might expect to have to use the premiss twice to close off this tree. So let’s press on, and instantiate (1) again

```
(10)            ¬Lnm
(11)        ∀y(∃z Lyz ⊃ Lny)           (from 1)
(12)        (∃z Lmz ⊃ Lnm)              √
             /           \
(13)    ¬∃z Lmz   √       Lnm
(14)    ∀z¬Lmz             *
(15)      ¬Lmn
           *
```

With a bit of effort, again we are done. The inference **E** is q-valid (and so valid).

It’s worth pausing just a moment to compare this last demonstration of validity with the simple informal proof in §5.1. Evidently, the tree proof is a *lot* less ‘natural’ and direct than the informal one.

But remember: trees are *not* being sold as the best way of formally mimicking ordinary styles of deduction (cf. §20.3): they are not a ‘natural deduction’ proof system. All that is being claimed is that trees are *one* well-motivated and reasonably manageable way of showing that inferences are q-valid when they are.

For yet more examples of our tree-rules at work, see Chapter 35.

29.3 Extending the (∀) rule

As we noted before (§28.3), our stated rules for tree-building certainly look *sound* (i.e. an argument warranted by a closed tree built following our rules is indeed q-valid). If we start a tree by assuming that $A_1, A_2, \ldots, A_n, \neg C$ are all true on some valuation *q*, then the quantifier rules for extending the tree are warranted by the principles (V1) to (V5) about valuations (§27.6). Hence, if the tree *does* end up in contradictions, there can be no such valuation *q*; and then the target inference $A_1, A_2, \ldots, A_n \therefore C$ must be q-valid. We'll turn this rough and ready proof-idea into a tidier affair in the next chapter.

But is our basic set of rules *complete* (i.e. do we have enough tree-building rules to find a closed tree corresponding to *every* q-valid inference)?

Consider the inference '∀xFx ∴ ∃xFx'. If we try to construct a tree, we'll get as far as

(1)	∀xFx	
(2)	¬∃xFx	√
(3)	∀x¬Fx	

Then we have to stop, as there is no further rule to apply. So the tree doesn't close. On the other hand, the inference *is* q-valid according to our account. Take any q-valuation *q*. If *q* makes '∀xFx' true, then every object in *q*'s domain needs to be in the extension of 'F'. Since domains are non-empty, that means at least one thing is in the extension of 'F'. So *q* must make '∃xFx' true too.

Now, this last point only holds because we chose to say that domains always have at least one member (§27.2). Suppose we'd taken the other tack and allowed empty domains. Then '∀xFx ∴ ∃xFx' would no longer be q-valid. Given an empty domain it is vacuously true that, if any object is in the domain, it is in the extension of 'F' (that's a general conditional claim with an antecedent which is always false). So '∀xFx' is true in an empty domain. But '∃xFx' will be false.

In sum, our tree-rules are currently out of harmony with our definition of q-validity. Our rules *don't* warrant the inference '∀xFx ∴ ∃xFx'; our definition of q-validity *does*. What to do?

There is a forking of the paths here. You might think the obvious response is to backtrack hastily, revise our definition of validity, adopting instead the idea of q_0-validity (defined in terms of q_0-valuations, which are exactly like q-valuations except that they allow empty domains). And if we do that, it can be proved that our current set of rules *would* then be complete. There are closed trees using our existing rules for all and only the q_0-valid arguments. So that looks like a happy restoration of harmony. Unfortunately, there is a snag.

Consider the inferences '∀xFx ∴ Fn' and ' Fn ∴ ∃xFx'. These are both straightforwardly q_0-valid (take any valuation of the vocabulary in the premisses and conclusion: that will involve assigning an object as q-value to 'n'; so *these* valuations won't have empty domains). Then we have '∀xFx ∴ Fn' and ' Fn ∴ ∃xFx' are q_0-valid and '∀xFx ∴ ∃xFx' isn't. Or putting it more starkly, we have *A* q_0-entails *B*, *B* q_0-entails *C*, but *A* does *not* q_0-entail *C*. Which goes clean against our normal assumption that entailments can be chained into longer arguments and our ultimate conclusion will still follow from our initial premiss(es). What to do?

Well, we could adopt a notion of entailment which allows exceptions to that fundamental assumption. Or we could be revisionary in another way and reject '∀xFx ∴ Fn' as a valid entailment in the absence of an explicit premiss to the effect 'n exists'. But most logicians prefer to stick to the two assumptions that we can unrestrictedly chain entailments and endorse '∀xFx ∴ Fn'.

We are going to follow the majority here (though see Chapter 36 for some more comments). Hence we shall *not* revise our definition of q-validity to allow empty domains. That means '∀xFx ∴ ∃xFx' remains q-valid. It also means, however, that we need a further tree-building rule if we are to be able to warrant every q-valid argument by means of a closed tree.

One popular option here is to relax the rule (∀) rather radically. The rule currently says that if *W* is of the form $\forall v C(\ldots v \ldots v \ldots)$, then we can add $C(\ldots c \ldots c \ldots)$ to any open path on which *W* appears, where *c* is a constant already on the path. Suppose instead that we are allowed to instantiate *W* with *any* constant, old or new. Then we can close the tree which we had trouble with before:

(1)	∀xFx	
(2)	¬∃xFx	√
(3)	∀x¬Fx	
(4)	Fc_0	
(5)	$\neg Fc_0$	
	✻	

Here we've just arbitrarily chosen to use the constant 'c_0'. Since we are assuming that there is at least one thing in the domain, pick such an object, and we can surely use a new name to dub it. Instantiating the universal quantifications at (1) and (3) with a name of this object then exposes a contradiction.

But our suggested generous rule rather ruins the memorable symmetry between our rules (∀) and (∃), summed up in the slogan 'instantiate universals with old constants, existentials with new ones'. There's something unattractive too about relaxing (∀) to allow us to instantiate universals with *any* constant, and then immediately having to issue the warning that you'd be daft ever to instantiate a universal with a random constant except in the very special circumstances that there are no other constants in sight to play with.

So rather than go too radical, here's our preferred revision of the rule (∀). We'll keep the old rule for when there *are* constants already on the relevant path, and just add a clause to deal with the special case when we want to instantiate a universal quantification and there are *no* preceding constants:

(∀′) *W* is of the form $\forall v C(\ldots v \ldots v \ldots)$: add $C(\ldots c \ldots c \ldots)$ to an open path on which *W* appears, where *c* is *either* a constant on that path (and which hasn't been used to instantiate *W* before), *or* is a new constant (and there are no other constants appearing on the path). *W* is *not* checked off.

$$\begin{array}{c} \forall v C(\ldots v \ldots v \ldots) \\ | \\ C(\ldots c \ldots c \ldots) \end{array} \quad [c \text{ old, or unprecedented}]$$

NB: Rule (∀′) still allows us to construct our mini-tree warranting ‘∀xFx ∴ ∃xFx’.

We will show in the next chapter that this minor extension of our original rule does indeed give us a sound and complete set of rules. *So for the rest of this book, we’ll assume the revised rule* (∀′) *is in play.*

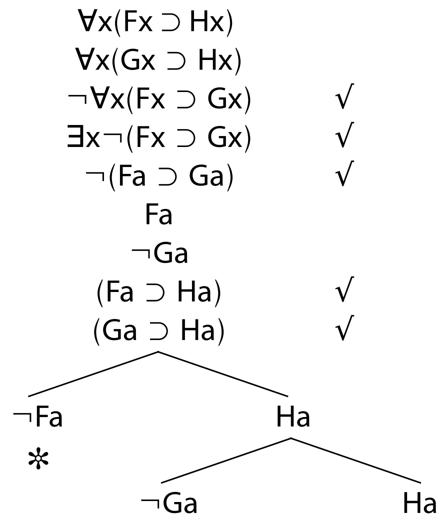
29.4 What can be learnt from open trees?

We have emphasized before that the fact that a quantifier tree doesn’t close doesn’t *always* show the argument under consideration is invalid. An injudicious sequence of choices of the rules to apply can mean that a tree trails off down an infinite path, when better choices would close off the tree (see §25.5 again).

However, in simple cases, a tree can terminate with no more rules left to apply, but leaving an open branch. If there *is* an open branch on such a completed tree, then we *can* read off from the tree a q-valuation which makes the premisses true and conclusion false and which thus demonstrates q-invalidity. Consider, for example,

E ∀x(Fx ⊃ Hx), ∀x(Gx ⊃ Hx) ∴ ∀x(Fx ⊃ Gx)

Starting in the obvious way, and omitting the unofficial line numbers, we get:



The tree has open branches. The only unchecked wffs which aren’t atomic wffs or negations of atomic wffs are the universal quantifications at the top of the tree. But we have no new names available to instantiate them with. So we can’t use the (∀′) rule again. There are no more rules to apply.

In the case of **PL** trees, if a branch stays open, we can read off a valuation which makes all the wffs on the branch true. *The trick is to pick a valuation which makes the primitives on the branch, i.e. the atoms and negated atoms, all true.* Then truth percolates up the branch. That makes the initial premisses and negated conclusion all true, and the valuation therefore shows that the inference is invalid (cf. §§16.2, 19.2). Let’s try the very same trick here.

In the present case, it doesn’t matter which open branch we choose, the primitives on the branch are ‘Fa’, ‘¬Ga’, and ‘Ha’. We need to choose a reference for the name ‘a’. Let’s assign it the first number, i.e., 0. And then, to give the primitives the right values, we need the extensions of ‘F’ and ‘H’ to *include* 0, and the extension of ‘G’ *not* to include 0.

A domain must of course always contain any named objects and/or objects which are involved in the extension of predicates. So in the present case, we know the domain must contain at least 0. What else should we put in the domain? *Nothing*. We don’t want any extra objects in the domain which we haven’t any information about (and which might therefore have unwanted effects on the truth-values of quantified wffs). So, keeping the domain as small as possible, we get:

> Domain = {0}, i.e. the set containing just 0.
> Reference of ‘a’ = 0.
> Extension of ‘F’ = extension of ‘H’ = {0}.
> Extension of ‘G’ = { }, i.e. the empty set.

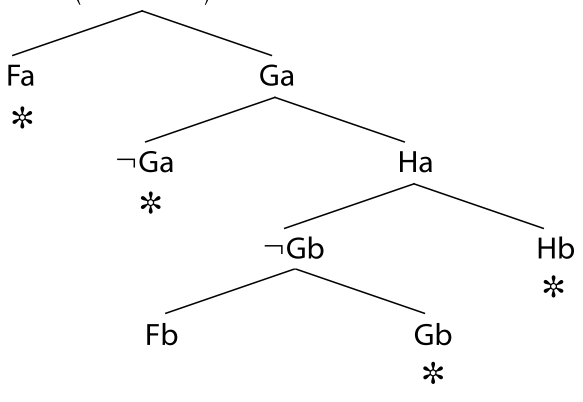
This makes the primitives all true. And it is readily checked that this indeed makes *all* the wffs on an open branch of our tree true; and in particular, it makes the premisses of **E** true and conclusion false (compare the discussion of the same argument in §28.5).

Here’s another example:

F $\forall x(Fx \lor Gx), \forall x(Gx \supset Hx) \therefore (\forall xFx \lor \forall xHx)$

Using our rules of thumb, we get the following (again omitting line numbers):

$\forall x(Fx \lor Gx)$
$\forall x(Gx \supset Hx)$
$\neg(\forall xFx \lor \forall xHx)$ √
$\neg\forall xFx$ √
$\neg \forall xHx$ √
$\exists x\neg Fx$ √
$\exists x\neg Hx$ √
$\neg Fa$
$\neg Hb$
$(Fa \lor Ga)$ √
$(Fb \lor Gb)$ √
$(Ga \supset Ha)$ √
$(Gb \supset Hb)$ √



Here, the initial steps are pretty automatic. Then we instantiate the two universal quantifications with every name in sight, and unpack the resulting wffs. There are no more rules to apply, and the tree stays open. So let's try to construct a countervaluation for the argument. Take the open path, where we have

¬Fa, Fb, Ga, ¬Gb, Ha, ¬Hb

There are two names here to which we need to assign q-values. Let's take them in order to refer to the first two numbers. Again we take the domain of discourse to be no larger than it needs to be to include those references for the names. The resulting valuation which makes the primitives all true is then:

Domain = {0, 1}
Reference of 'a' = 0
Reference of 'b' = 1
Extension of 'F' = {1}
Extension of 'G' = {0}
Extension of 'H' = {0}

It is easily checked again that this valuation makes all the wffs on the open branches true. In particular, it makes the premisses of **F** true and conclusion false. Why does it make, e.g., the universal quantification (1) true? Because it makes an instance true for each object in the domain (to emphasize, there are no more objects in the domain than the named ones).

When does this trick work, i.e. this trick of reading valuations from the primitives on an open branch on a **QL** tree and arriving at a countervaluation for the argument in question? For a start, it works for arguments where the propositions don't feature quantifiers embedded one inside the other. Instantiate all the existential quantifiers, thereby introducing a collection of new names; instantiate all the universal quantifications with respect to all the names thereby introduced (plus any others in the original premisses and conclusion). And if the tree doesn't close, construct the 'smallest domain' valuation which contains one object per name and which makes the primitives on some open branch true, and you'll get the desired countermodel.

We saw, however, that when an argument involves embedded quantifiers life gets more complicated. It may be the case that the only countermodels which show that the inference is invalid are infinite ones; see the discussion of §28.5 **G**. Yet it may also be the case that, when we *do* find an infinite branch that never closes, it doesn't show that the argument is invalid, and we can't read off a genuine countervaluation from the branch; see again the discussion of the example in §25.5. Looking more closely at the infinite open branch in that last case, we see that it in fact was constructed by ignoring its lines (3) and (5). No surprise then that choosing a valuation that makes all the primitives on that branch true won't make (3) and (5) true too! We can only form countermodels from infinite branches when the trees have been constructed more *systematically*, without ignoring any complex wffs on an open branch; but that's not entirely straightforward (more on this in the next chapter).

29.5 Summary

- We summarized our original set of tree-building rules for quantifier trees, and argued for a modification of the rule (∀) to match our restriction to non-empty domains.
- A couple of useful heuristic tips – not strict rules, but guides for applying the rules in a sensible order: *eliminate negated quantifiers first* and then *instantiate existentials before universals*.
- In some simple cases at any rate, we can read countervaluations from open trees (choose a 'smallest domain' valuation that makes the primitives on an open branch true).

Exercises 29

A Use **QL** trees to evaluate the entailment claims (1) to (10) in Exercises **28A**.

B Using trees, show the following arguments are valid:

1. Some philosophers admire Jacques. No one who admires Jacques is a good logician. So some philosophers are not good logicians.
2. Some philosophy students admire all logicians; no philosophy student admires any rotten lecturer; hence, no logician is a rotten lecturer.
3. There's a town to which all roads lead. So all roads lead to a town.
4. Some good philosophers admire Frank; all wise people admire any good philosopher; Frank is wise; hence there is someone who both admires and is admired by Frank.
5. Any true philosopher admires some logician. Some students admire only existentialists. No existentialists are logicians. Therefore not all students are true philosophers.
6. Everyone loves a lover; hence if someone is a lover, everyone loves everyone!
7. If anyone speaks to anyone, then someone introduces them; no one introduces anyone to anyone unless they know them both; everyone speaks to Frank; therefore everyone is introduced to Frank by someone who knows him. [*Use* 'Rxyz' to render '*x introduces y to z*'.]
8. Any elephant weighs more than any horse. Some horse weighs more than any donkey. If a first thing weighs more than a second thing, and the second thing weighs more than a third, then the first weighs more than the third. Hence any elephant weighs more than any donkey.

C Redo the first three examples of §29.2 as *signed* trees (as in §28.2).

35

Trees for identity

We now expand our treatment of quantifier trees to cover arguments involving identity, expressed in $\mathbf{QL}^{=}$. We first introduce and motivate two rules dealing with wffs involving identity. We explore some introductory examples and also show that these rules can be used to demonstrate that the basic laws of identity are q-logical truths. Then we consider some examples involving definite descriptions and numerical quantifiers.

35.1 Leibniz's Law again

Take the argument: 'Bertie is clever. Bertie is none other than Russell. So Russell is clever.' That's valid. Translating, we have

A $\quad$ Fm, m = n ∴ Fn

Starting a signed tree in the standard way, we suppose that there is a valuation q such that

(1) $\quad \mathsf{Fm} \Rightarrow_q \mathrm{T}$
(2) $\quad \mathsf{m = n} \Rightarrow_q \mathrm{T}$
(3) $\quad \neg\mathsf{Fn} \Rightarrow_q \mathrm{T}$

How are we to proceed now? Consider again Leibniz's Law, in the form we met in §32.5:

(LL′) If '*a*' and '*b*' are functioning as co-referential designators in the claims *C(…a…a…)* and *C(…b…b…)*, and also the context *C* expresses the same property in each case, then these claims say the same about the same thing, and so must have the same truth-value.

We noted that, in English, different occurrences of the same predicate need not attribute the same property (remember the behaviour of '… is so-called because of his size'). But in our tidy artificial language $\mathbf{QL}^{=}$ we just don't have that troublesome kind of predicate. And our official semantics indeed implies

(A) If 'm' and 'n' are co-referential designators according to q, then the claims 'Fm' and 'Fn' must have the same truth-value on q.

Further, it is immediate that

(B) The atomic wff 'm = n' is true on the valuation q just if 'm' and 'n' are assigned the same q-values by q.

(That follows from ($Q0^+$) in §33.1.) Putting (A) and (B) together, they imply

(C) If the claims 'Fm' and 'm = n' are true on the valuation q, then 'Fn' must also be true on q.

Hence, given (1) and (2), we can appeal to (C) and continue our tree-argument

(4) $\mathsf{Fn} \Rightarrow_q \mathrm{T}$

which immediately contradicts (3), and our tree closes.

We can evidently generalize the (A)–(B)–(C) reasoning; and this warrants the following corresponding rule for building unsigned trees:

(L) If W_1 is of the form $C(\ldots m \ldots m \ldots)$, involving one or more occurrences of the constant m, and W_2 is of either the form $m = n$ or the form $n = m$, then we can add $C(\ldots n \ldots n \ldots)$, formed by substituting some or all of the occurrences of m in W_1 by occurrences of n, to the foot of any open branch containing both W_1 and W_2. Schematically,

$$\begin{array}{c} C(m) \\ m = n \text{ or } n = m \\ | \\ C(n) \end{array}$$

Two comments on this:

- (L) differs from all our previous rules for tree-building in taking *two* wffs as input. The inputs can appear, of course, in either order earlier on the branch. Since the conclusion added to the branch does not exhaust the combined contents of the input wffs, we *don't* check them off.
- Note that previous rules involving substitution – the universal quantifier instantiation rule (∀), the existential instantiation rule (∃) – tell us to uniformly replace *every* occurrence of one symbol (a variable) with another symbol (a constant). By contrast, our rule (L) can be more permissive. It allows us to replace one or more occurrences of one constant with another. So take, for example, the wff

  ∀x(Raxa ⊃ Fa)

  We can focus on the middle occurrence of 'a', and take all '∀x(Rax_⊃ Fa)' to be 'context'. Then, given 'a = b' and applying rule (L), we can infer

  ∀x(Raxb ⊃ Fa)

  Alternatively, we can focus on the first and last 'a's, take the context to be '∀x(R_xa ⊃ F_)', and use (L) to infer

  ∀x(Rbxa ⊃ Fb).

  Or, of course, we can infer the result of simultaneously replacing all three occurrences of 'a'. Each inference is plain valid.

So now let's take some simple examples of our rule (L) in operation.

First, consider the argument

B Either Alice or Mrs Brown is the winner. Alice isn't the winner. So Alice isn't Mrs Brown.

It is trivial to render this into **QL⁼**, and the resulting tree is equally straightforward:

```
(1)                 (Fa ∨ Fb)      √
(2)                   ¬Fa
(3)                 ¬¬a = b        √
(4)                   a = b
                  /         \
(5)             Fa           Fb
(6)             ✻            Fa
                             ✻
```

Here, the only step worth remarking on is (6), where we have used (L) to derive 'Fa' from (5) 'Fb' and (4) 'a = b'. The tree closes, hence the intuitively valid argument (B) is, as we'd expect, q-valid.

For another example, consider the following:

C Only Russell is a great philosopher. Bertrand is a great philosopher who smokes. Hence Russell smokes.

Put 'F' for *great philosopher*, 'G' for *smokes*, etc., and we can render this:

$$(Fm \wedge \forall x(Fx \supset x = m)),\ (Fn \wedge Gn)\ \therefore\ Gm$$

A tree can then be constructed as follows:

```
(1)            (Fm ∧ ∀x(Fx ⊃ x = m))      √
(2)                 (Fn ∧ Gn)             √
(3)                   ¬Gm
(4)                   Fm
(5)               ∀x(Fx ⊃ x = m)
(6)                   Fn
(7)                   Gn
(8)               (Fn ⊃ n = m)            √
                  /          \
(9)            ¬Fn            n = m
(10)            ✻              Gm          (by (L), from 7, 9)
                               ✻
```

(Check that you follow each step.) Again we have a valid argument.

What about the simple inference

D George is Mr Orwell; Mr Orwell is Eric Blair; hence George is Eric Blair?

With the obvious translation scheme, the tree for this inference is

(1)	$m = n$		
(2)	$n = o$		
(3)	$\neg m = o$		
(4)	$m = o$		(by (L), from 1, 2)
	✻		

To see that the move to (4) is warranted, think of '$m = n$' as of the form $C(n)$, with the context C being '$m = \_$'; and then from $C(n)$ and '$n = o$' we infer the corresponding $C(o)$, i.e. '$m = o$'.

Generalizing that last example, recall the transitivity law

(Tran) $\forall x \forall y \forall z((x = y \wedge y = z) \supset x = z)$

This is, as we'd expect, a q-logical truth. But how can we use a tree to show this? We start the tree with the negation of (Tran) and aim for absurdity.

(1)	$\neg \forall x \forall y \forall z((x = y \wedge y = z) \supset x = z)$	√	
(2)	$\exists x \neg \forall y \forall z((x = y \wedge y = z) \supset x = z)$	√	
(3)	$\neg \forall y \forall z((m = y \wedge y = z) \supset m = z)$	√	
(4)	$\exists y \neg \forall z((m = y \wedge y = z) \supset m = z)$	√	
(5)	$\neg \forall z((m = n \wedge n = z) \supset m = z)$	√	
(6)	$\exists z \neg((m = n \wedge n = z) \supset m = z)$	√	
(7)	$\neg((m = n \wedge n = o) \supset m = o)$	√	
(8)	$(m = n \wedge n = o)$	√	
(9)	$\neg m = o$		
(10)	$m = n$		
(11)	$n = o$		
(12)	$m = o$		(by (L), from 10, 11)
	✻		

The moves up to line (11) are all automatic. We drive in (1)'s initial negation sign to get (2), where we can do nothing else but instantiate the existential quantifier to get (3). That pattern repeats twice again until we get to (8), when we apply the familiar connective rules to get us to (9), (10) and (11). The final step is exactly as in the previous tree.

Next, consider the following inference:

E Only Angharad and Bryn love Mrs Jones; someone who loves Mrs Jones speaks Welsh; so either Angharad or Bryn speaks Welsh.

For simplicity, let's use the simple 'Gx' to render '*x* loves Mrs Jones' (for the internal structure of '… loves Mrs Jones' isn't actually used in the argument, so we needn't explicitly represent it). Then the argument goes into **QL$^=$** as

$$((Ga \wedge Gb) \wedge \forall x(Gx \supset (x = a \vee x = b))),\ \exists x(Gx \wedge Fx)\ \therefore\ (Fa \vee Fb)$$

The corresponding closed tree starts more or less automatically

(1)	$((Ga \wedge Gb) \wedge \forall x(Gx \supset (x = a \vee x = b)))$	√
(2)	$\exists x(Gx \wedge Fx)$	
(3)	$\neg(Fa \vee Fb)$	√
(4)	$(Ga \wedge Gb)$	

```
(5)          ∀x(Gx ⊃ (x = a ∨ x = b))
(6)                   ¬Fa
(7)                   ¬Fb
```

Next, following our rule of thumb 'instantiate-existentials-before-universals', we check off (2) and instantiate with a new name to continue as follows:

```
(8)                 (Gc ∧ Fc)                    √
(9)                    Gc
(10)                   Fc
(11)          (Gc ⊃ (c = a ∨ c = b))             √
                 /              \
(12)          ¬Gc          (c = a ∨ c = b)       √
               *             /          \
(13)                      c = a        c = b
(14)                       Fa           Fb
                           *            *
```

Here, on the left branch, 'Fa' is derived from (10) and (13); similarly for 'Fb' on the right branch.

Finally in this section, let's partially fulfil the promise we made in §33.3 to show that the following wffs are equivalent:

F ∃x(Fx ∧ ∀y(Fy ⊃ y = x))
G ∃x∀y(Fy ≡ y = x)

We will give a tree to show that **F** q-entails **G**:

```
(1)          ∃x(Fx ∧ ∀y(Fy ⊃ y = x))      √
(2)             ¬∃x∀y(Fy ≡ y = x)          √
(3)             ∀x¬∀y(Fy ≡ y = x)
(4)           (Fa ∧ ∀y(Fy ⊃ y = a))        √
(5)                     Fa
(6)                ∀y(Fy ⊃ y = a)
```

So far, so automatic (we dealt with the negated quantifier at (2), and then instantiated the only existential quantifier). We now instantiate (3) with the name 'a':

```
(7)                ¬∀y(Fy ≡ y = a)         √
(8)                ∃y¬(Fy ≡ y = a)         √
(9)                 ¬(Fb ≡ b = a)          √
(10)                 (Fb ⊃ b = a)          √          (from 6)
                   /                \
(11)          ¬Fb                      b = a
             /     \                  /     \
(12)      Fb        ¬Fb            Fb        ¬Fb    } (from 9)
(13)    ¬b = a     b = a         ¬b = a     b = a   }
(14)      *         ¬Fa            *         ¬Fa
(15)                 *                        *
```

We use (L) at line (14) to derive '¬Fa' from the previous two wffs on the path: the other paths close off without appealing to (L). We will leave it as an exercise to show that, conversely, **G** q-entails **F**.

35.2 Self-identity

Consider next the law of self-identity

(Ref) $\forall x\, x = x$

This is again a q-logical truth. How can we construct a tree to show this? As usual, by assuming it is false on an evaluation and deriving an absurdity. Commencing the tree, we get:

(1)	$\neg\forall x\, x = x$	√
(2)	$\exists x \neg x = x$	√
(3)	$\neg a = a$	

Now, we haven't yet exposed a formal contradiction in the sense of a pair of wffs of the form *W* and ¬*W*. Still, getting to something of the form ¬*n* = *n* is surely just as bad! For plainly this is itself an outright logical absurdity. Nothing can possibly fail to be identical to itself. *So let's adopt a new rule for closing branches, which treats denials of self-identities on a par with contradictions.*

Henceforth, then, our rules for **QL**$^=$ trees will be the **QL** rules, together with the new rule (L) for extending trees, *and now also our new rule for closing branches* – so we need to replace the structural rule (2) given in §29.1 with this:

(2$^=$) Inspect any not-yet-closed path down from the top of the tree to see whether it involves either an explicit contradiction (i.e. for some *W*, it contains both *W* and also the corresponding formula ¬*W*), or a wff of the form ¬*n* = *n*. If it does, then we *close* that path with a star as an absurdity marker, and ignore it henceforth.

With our last new rule in place, we can also prove e.g. that identity obeys the symmetry law

(Sym) $\forall x \forall y(x = y \supset y = x)$.

For consider the following tree:

(1)	$\neg\forall x\forall y\,(x = y \supset y = x)$	√
(2)	$\exists x\neg\forall y\,(x = y \supset y = x)$	√
(3)	$\neg\forall y\,(a = y \supset y = a)$	√
(4)	$\exists y\neg(a = y \supset y = a)$	√
(5)	$\neg(a = b \supset b = a)$	√
(6)	$a = b$	
(7)	$\neg b = a$	
(8)	$\neg b = b$	(by (L) from 6, 7)
	✻	

Here, think of '¬b = a' at (7) as being formed from a frame C (i.e. '¬b = _') filled by ‘a’, and then from C(a) and ‘a = b’ we get C(b) by Leibniz’s Law. We can close the tree as we’ve arrived at an absurd denial of self-identity.

Two comments on this last proof. First, note that the wffs at lines (6) and (7), i.e. ‘a = b’ and ‘¬b = a’ are *not* a formal contradiction: for they are not strictly a pair of the form W and $\neg W$. Second, we *could* have introduced a symmetry rule for identity, allowing us to move from a wff of the form $m = n$ to one of the corresponding wff $n = m$. With that rule in play, we could have completed the last tree by applying the symmetry rule to (6) to derive

```
(8′)                b = a
                      ✻
```

and the tree would have closed off without appeal to our new closure rule. But with our new closure rule in play, an explicit symmetry rule would be redundant.

For another simple example, take the argument

H Only Bertrand is a great philosopher. Russell is a great philosopher. Hence Bertrand is Russell.

With an obvious rendition of the argument into **QL⁼**, the tree starts

```
(1)          (Fm ∧ ∀x(Fx ⊃ x = m))       √
(2)                   Fn
(3)                  ¬m = n               √
(4)                   Fm
(5)               ∀x(Fx ⊃ x = m)
(6)                (Fn ⊃ n = m)           √
                  /            \
(7)           ¬Fn               n = m
(8)            ✻               ¬m = m    (by (L) from 3, 7)
                                  ✻
```

Here we have completed the right-hand branch by dancing the same kind of two-step as before – i.e. we have treated ‘¬m = n’ as C(n), and used ‘n = m’ to infer the corresponding C(m), and then used our new structural principle for closing trees.

35.3 Descriptions again

In this section, we’ll consider some arguments involving the use of Russell’s theory of descriptions. Consider firstly

I The author of the Iliad wrote the Odyssey. Homer wrote the Iliad. Hence, Homer wrote the Odyssey.

Taking ‘the author of the Iliad’ to mean ‘the person who wrote the Iliad’, this is a valid argument.

Let’s use ‘F’ to render ‘wrote the Iliad’, ‘G’ for ‘wrote the Odyssey’. Then the argument can be rendered

$$\exists x((Fx \wedge \forall y(Fy \supset y = x)) \wedge Gx),\ Fm\ \therefore\ Gm$$

And the corresponding tree is

```
(1)            ∃x((Fx ∧ ∀y(Fy ⊃ y = x)) ∧ Gx)   √
(2)                         Fm
(3)                        ¬Gm
(4)              ((Fa ∧ ∀y(Fy ⊃ y = a)) ∧ Ga)   √
(5)                (Fa ∧ ∀y(Fy ⊃ y = a))        √
(6)                         Ga
(7)                         Fa
(8)                   ∀y(Fy ⊃ y = a)
(9)                    (Fm ⊃ m = a)             √
                       /          \
(10)                ¬Fm            m = a
(11)                 *             ¬Ga           (by (L) from 3, 10)
                                    *
```

Note, by the way, that had we rendered the designator ‘the author of the Iliad’ by a simple unstructured name ‘n’ – so the first premiss is translated ‘Fn’ – we wouldn’t have been able to prove validity.

Another example:

J Homer, and no-one else, wrote the Iliad. Homer wrote the Odyssey. Hence, the author of the Iliad wrote the Odyssey.

Translating, using the same key, we can start the tree

```
(1)               (Fm ∧ ∀x(Fx ⊃ x = m))
(2)                         Gm
(3)            ¬∃x((Fx ∧ ∀y(Fy ⊃ y = x)) ∧ Gx)
```

Unpacking (1) and eliminating the negated quantifier in (3), we immediately get

```
(4)                         Fm
(5)                   ∀x(Fx ⊃ x = m)
(6)            ∀x¬((Fx ∧ ∀y(Fy ⊃ y = x)) ∧ Gx)
```

Instantiating the universal quantification in (6) with ‘m’ – what else? – we naturally continue

```
(7)                ¬((Fm ∧ ∀y(Fy ⊃ y = m)) ∧ Gm))      √
                              /           \
(8)          ¬(Fm ∧ ∀y(Fy ⊃ y = m))  √      ¬Gm
                    /          \              *
(9)          ¬Fm            ¬∀y(Fy ⊃ y = m)
(10)          *
```

Now, we’ve almost got an explicit contradiction between the right-hand wff at (9) and (5): but not quite. ‘∀x(Fx ⊃ x = m)’ and ‘¬∀y(Fy ⊃ y = m)’ involve different variables and aren’t strictly speaking a pair of the form *W* and ¬*W*. If we’d had the foresight to render the initial premiss as ‘(Fm ∧ ∀y(Fy ⊃ y = m))’ – which would have been quite legitimate – then we *would* have had an explicit contradiction by line (9). As things are, we have to continue as follows:

(10)	$\exists y\neg(Fy \supset y = m)$	(from 9).
(11)	$\neg(Fa \supset a = m)$	

Instantiating (5) we have

(12)	$(Fa \supset a = m)$
	✻

And we are done. Note that, in this last case, we *haven't* had to use the special rules for identity to show this argument is valid. This is sometimes the way: use of the identity relation may be naturally involved in the translation of the premisses and conclusion of an argument; but it may so happen that showing the resulting inference is q-valid only uses the fact that it is a relation and not the further special logical properties of this relation.

Here, though, is another case where we do have to appeal to the special laws of identity. Consider

K The author of the Iliad wrote the Odyssey. Hence at most one person wrote the Iliad.

According to Russell's Theory of Descriptions, the premiss is only true if there is a unique author, and is false if there is more than one author. So, on Russell's view, the argument is a valid one. Well, be that as it may about the vernacular argument, its closest rendition into **QL**$^=$ is certainly q-valid. For consider:

(1)	$\exists x((Fx \wedge \forall y(Fy \supset y = x)) \wedge Gx)$	
(2)	$\neg\forall x\forall y((Fx \wedge Fy) \supset x = y)$	√
(3)	$\exists x\neg\forall y((Fx \wedge Fy) \supset x = y)$	√
(4)	$\neg\forall y((Fa \wedge Fy) \supset a = y)$	√
(5)	$\exists x\neg((Fa \wedge Fy) \supset a = y)$	√
(6)	$\neg((Fa \wedge Fb) \supset a = b)$	√
(7)	$(Fa \wedge Fb)$	
(8)	$\neg a = b$	
(9)	Fa	
(10)	Fb	

So far, we've unpacked (2) in automatic steps. Let's now instantiate (1), using yet another new name, check it off and unpack the resulting triple conjunction:

(11)	$((Fc \wedge \forall y(Fy \supset y = c)) \wedge Gc)$	√
(12)	$(Fc \wedge \forall y(Fy \supset y = c))$	√
(13)	Gc	
(14)	Fc	
(15)	$\forall y(Fy \supset y = c)$	

Where now? Well, we plainly need to extract information from this last universal quantification. It would be pretty pointless to instantiate it with respect to 'c' (why?). But we have two other names to play with, yielding

```
(16)                 (Fa ⊃ a = c)              √
(17)                 (Fb ⊃ b = c)              √
                    /            \
(18)            ¬Fa               a = c
                 *               /     \
(19)                          ¬Fb       b = c
                               *
```

The right-hand branch is still open: what is there left to make use of? Well, note that at (8) we have '¬a = b', and now on the same branch we have both 'a = c', and 'b = c', and those wffs are clearly an inconsistent triad. Take 'a = c' as *C*(c), and use the 'b = c' to derive *C*(b) by (L), and the branch closes thus ...

```
(20)                                    a = b
                                          *
```

We can similarly, and rather more simply, show that the following two arguments are also valid:

L The author of the Iliad wrote the Odyssey. Hence at least one person wrote the Iliad.

M The author of the Iliad wrote the Odyssey. Hence whoever wrote the Iliad wrote the Odyssey.

Those cases are left as exercises. But we will demonstrate the validity of

N A least one person wrote the Iliad. At most one person wrote the Iliad. Whoever wrote the Iliad wrote the Odyssey. Hence, the author of the Iliad wrote the Odyssey.

The tree starts straightforwardly enough, following our familiar rules of thumb:

```
(1)                         ∃xFx                          √
(2)                 ∀x∀y((Fx ∧ Fy) ⊃ x = y)
(3)                       ∀x(Fx ⊃ Gx)
(4)              ¬∃x((Fx ∧ ∀y(Fy ⊃ y = x)) ∧ Gx)          √
(5)              ∀x¬((Fx ∧ ∀y(Fy ⊃ y = x)) ∧ Gx)
(6)                          Fa
(7)                ¬((Fa ∧ ∀y(Fy ⊃ y = a)) ∧ Ga)          √
                           /            \
(8)          ¬(Fa ∧ ∀y(Fy ⊃ y = a))        ¬Ga
```

So far, so automatic. The right hand branch quickly closes, for we have

```
(9)                 |             (Fa ⊃ Ga)        (from 3)
                    |            /         \
(10)                |         ¬Fa           Ga
                    |          *             *
```

Meanwhile, the left-hand branch can be continued from (8) like this:

(9)	$\neg Fa$	$\neg\forall y(Fy \supset y = a)$	√	(from 8)
(10)	✻	$\exists y\neg(Fy \supset y = a)$	√	
(11)		$\neg(Fb \supset b = a)$	√	
(12)		Fb		
(13)		$\neg b = a$		

Inspection reveals that we haven't yet made any appeal to (2). So let's instantiate it with respect to the two names in play, thus:

(14)		$\forall y((Fb \wedge Fy) \supset b = y)$			(from 2)
(15)		$((Fb \wedge Fa) \supset b = a)$			√
(16)		$\neg(Fb \wedge Fa)$		$b = a$	
				✻	
(17)	$\neg Fb$		$\neg Fa$		
	✻		✻		

Once more, we arrive at a closed tree.

Note, by the way, that we could put together the proofs for examples **K** to **M** to show that

(R) $\exists x((Fx \wedge \forall y(Fy \supset y = x)) \wedge Gx)$

entails each of the conjuncts in the more long-winded triple conjunction

$$(\{\exists xFx \wedge \forall x\forall y((Fx \wedge Fy) \supset x = y)\} \wedge \forall x(Fx \supset Gy))$$

And the proof for **N** shows that this triple conjunction entails (R) – thus justifying the claim (§34.1) that these two forms of Russell's Theory of Descriptions for rendering 'The *F* is *G*' into **QL**$^=$ are equivalent.

35.4 'One and one make two'

Finally in this chapter, consider the following inference:

O $\exists x(Fx \wedge \forall y(Fy \supset y = x)), \exists x(Gx \wedge \forall y(Gy \supset y = x)), \neg\exists x(Fx \wedge Gx)$
$\therefore \exists x\exists y(([\{Fx \vee Gx\} \wedge \{Fy \vee Gy\}] \wedge \neg x = y) \wedge$
$\forall z[\{Fz \vee Gz\} \supset \{z = x \vee z = y\}])$

which renders 'There is exactly one *F*, there is exactly one *G*, nothing is both *F* and *G*, so there are exactly two things which are *F* or *G*' (i.e. one thing and another thing make two altogether).

This inference is provably q-valid. To warrant **O** via a tree requires a little effort, but it is fun (of a kind) to show that it can be done. We start:

(1) $\exists x(Fx \wedge \forall y(Fy \supset y = x))$
(2) $\exists x(Gx \wedge \forall y(Gy \supset y = x))$
(3) $\neg\exists x(Fx \wedge Gx)$
(4) $\neg\exists x\exists y(([\{Fx \vee Gx\} \wedge \{Fy \vee Gy\}] \wedge \neg x = y) \wedge$
$\forall z[\{Fz \vee Gz\} \supset \{z = x \vee z = y\}])$

First, then, eliminate the two negated quantifiers, and then instantiate the two existential quantifiers (what else?):

```
(5)                          ∀x¬(Fx ∧ Gx)
(6)     ∀x¬∃y(([{Fx ∨ Gx} ∧ {Fy ∨ Gy}] ∧ ¬x = y) ∧
                                   ∀z[{Fz ∨ Gz} ⊃ {z = x ∨ z = y}])
(7)                       (Fa ∧ ∀y(Fy ⊃ y = a))
(8)                       (Gb ∧ ∀y(Gy ⊃ y = b))
```

Now we might as well unpack the conjunctions in (7) and (8):

```
(9)                                Fa
(10)                         ∀y(Fy ⊃ y = a)
(11)                               Gb
(12)                         ∀y(Gy ⊃ y = b)
```

Next, instantiate (6) with a name, and deal with the resulting negated wff.

```
(13) ¬∃y(([{Fa ∨ Ga} ∧ {Fy ∨ Gy}] ∧ ¬a = y) ∧ ∀z[{Fz ∨ Gz} ⊃ {z = a ∨ z = y}])
(14) ∀y¬(([{Fa ∨ Ga} ∧ {Fy ∨ Gy}] ∧ ¬a = y) ∧ ∀z[{Fz ∨ Gz} ⊃ {z = a ∨ z = y}])
(15) ¬(([{Fa ∨ Ga} ∧ {Fb ∨ Gb}] ∧ ¬a = b) ∧ ∀z[{Fz ∨ Gz} ⊃ {z = a ∨ z = b}])
```

We now have to disentangle (15), which is a negated conjunction.

```
                                 /                                \
(16) ¬([{Fa ∨ Ga} ∧ {Fb ∨ Gb}] ∧ ¬a = b)    ¬∀z[{Fz ∨ Gz} ⊃ {z = a ∨ z = b}]
                        /                  \                     |
(17) ¬[{Fa ∨ Ga} ∧ {Fb ∨ Gb}]            ¬¬a = b                 |
               /            \               |                    |
(18)  ¬{Fa ∨ Ga}       ¬{Fb ∨ Gb}           |                    |
(19)      ¬Fa              ¬Fb              |                    |
(20)      ¬Ga              ¬Gb              |                    |
           ✻                ✻               |                    |
```

Continuing the middle branch at (17):

```
(18′)                                     a = b                  |
(19′)                                      Ga                    from 11, 18′
(20′)                                  ¬(Fa ∧ Ga)                from 5
                                      /          \               |
(21′)                              ¬Fa           ¬Ga             |
                                    ✻             ✻              |
```

And on the rightmost branch, we obviously have to continue like this:

```
(17″)                                ∃z¬[{Fz ∨ Gz} ⊃ {z = a ∨ z = b}]
(18″)                               ¬[{Fm ∨ Gm} ⊃ {m = a ∨ m = b}]
(19″)                                          {Fm ∨ Gm}
(20″)                                       ¬{m = a ∨ m = b}
(21″)                                            ¬ m = a
(22)                                             ¬m = b
                                               /          \
(23)                                         Fm            Gm
```

So far, in fact, this tree has almost written itself. We've applied the obvious building moves at each step. But now, checking back, we note that we haven't yet made use of the two universal quantifications at (10) and (12). Instantiating (10), we can quickly close off the left of the two remaining branches:

```
(24)                      (Fm ⊃ m = a)              |
                          /          \              |
(25)                   ¬Fm          m = a           |
                        *             *             |
```

We finish by instantiating (12):

```
(24′)                                         (Gm ⊃ m = b)
                                              /          \
(25′)                                      ¬Gm          m = b
                                            *             *
```

Phew! So the arithmetic inference **O** indeed turns out to be a piece of pure logic.

And that is probably *quite* enough by way of examples of (closed) $\mathbf{QL}^=$ trees! As for open trees, the remarks about **QL** open trees in §29.3 carry over with only minor changes: to read off countermodels from finished trees with open branches, we just must make sure that whenever we have a wff of the form $m = n$ on the open branch, the names involved are given the same object as q-value. But we won't delay over this.

35.5 Soundness and completeness again

Finally, we should add for the record that there is the predictable Basic Result about $\mathbf{QL}^=$ trees too. Our collection of rules is sound and complete, so that a closed $\mathbf{QL}^=$ tree correctly indicates a q-valid argument, and there is a closed $\mathbf{QL}^=$ tree corresponding to every q-valid argument involving identity.

To show this, we essentially need to re-run the arguments of Chapter 30. But we do need a few new tricks to handle identity, which make things just a bit more complicated. We'll cut ourselves some slack and won't give the details here.

35.6 Summary

- We have introduced two additional rules for quantifier tree-building, to govern trees with identity, namely a version of Leibniz's Law for extending trees, and a new rule for closing a branch when it contains an absurdity of the form $\neg n = n$.
- These two additional rules allow us to warrant valid $\mathbf{QL}^=$ inferences by the tree method, including inferences where Russell's Theory of Descriptions is used to render propositions, and inferences using numerical quantifiers.
- Our rules for $\mathbf{QL}^=$ trees give us not only a sound but also a complete proof-system for establishing q-validity.

Exercises 35

A Use $\mathbf{QL}^{=}$ trees to show the following inferences are valid:

1. Jack is Fingers. Fingers is never caught. Whoever is never caught escapes justice. So Jack escapes justice.
2. There is a wise philosopher. There is a philosopher who isn't wise. So there are at least two philosophers.
3. Whoever stole the goods, knew the safe combination. *Someone* stole the goods, and only Jack knew the safe combination. Hence Jack stole the goods.
4. For every number, there's a larger one. No number is larger than itself. So for every number, there's a distinct number which is larger than it.
5. No one who isn't Bryn loves Angharad. At least one person loves Angharad. So Bryn loves Angharad.
6. Exactly one person admires Frank. All and only those who admire Frank love him. Hence exactly one person loves Frank.
7. The present King of France is bald. Bald men are sexy. Hence whoever is a present King of France is sexy.
8. Angharad loves only Bryn and Caradoc, who are different people. So Angharad loves exactly two people.
9. Juliet kisses exactly two philosophers. Hence there is more than one philosopher.

B Show that the following wffs are q-logical truths

1. $\forall x \forall y(x = y \supset (Fx \supset Fy))$
2. $\forall y \forall z(y = z \supset (\forall x(Lxy \wedge Fy) \supset \forall x(Lxz \wedge Fz)))$

Thinking about the structure of those proofs, conclude that each way of filling out the following schema from §33.1 does indeed yield a q-logical truth:

(LS) $\forall v \forall w(v = w \supset (C(\ldots v \ldots v \ldots) \supset C(\ldots w \ldots w \ldots)))$

C Returning to the labelled wffs and inferences earlier in this chapter, show that

1. **G** q-entails **F**.
2. **L** is q-valid.
3. **M** is q-valid.

Finally, confirm that the two Russellian ways of translating *The F is G*, via the schemas (R) and (R′) of §34.1, are indeed equivalent, by showing

4. '$\exists x \forall y((Fy \equiv y = x) \wedge Gx)$' q-entails '$\exists x((Fx \wedge \forall y(Fy \supset y = x)) \wedge Gx)$'.
5. '$\exists x((Fx \wedge \forall y(Fy \supset y = x)) \wedge Gx)$' q-entails '$\exists x \forall y((Fy \equiv y = x) \wedge Gx)$'.