23 Translations

This chapter is mostly about formalizing, paraphrasing, rendering or transcribing English into QL languages – or, as I prefer to describe it, translating as best we can.

As we have seen, QL languages use a quantifier/variable notation to express generality. Within a given language, all quantifiers run over the same single domain. And the logical vocabulary of these languages is austerely sparse (in particular, we have just two quantifiers to play with, and we have no specially-designed device for expressing restricted quantifiers but have to use connectives to do the work).

The use of the quantifier/variable notation for generality is, of course, the strikingly novel feature of our new languages. However, this is modelled on familiar devices found in ordinary English and/or in mathematician’s English (namely prefixed quantifiers tied to pronouns, with ‘variables’ serving as pronouns): so the basics of the notation should actually not be hard to grasp. What will cause most of the translational headaches, as we will soon see, are the further requirements that we use a single domain, restrict quantifiers using connectives, and manage with just two quantifiers.

23.1 Restricted quantifiers again

(a) Take the typical situation where we need to explicitly restrict some of our quantifiers. Suppose, for example, we are working in a language like QL₁ where the generic quantifiers range inclusively over all people, past and present. Then to render the two propositions

(1) All logicians are wise,
(2) Some logicians are wise,

we need to restrict our quantifiers using connectives, like this:

(3) \( \forall x (Gx \rightarrow Hx) \),
(4) \( \exists x (Gx \land Hx) \).

The translations of (1) using a conditional and (2) using a conjunction are the intuitively plausible options. But let’s now pause to emphasize why the connectives do have to go this way round. So note:

(i) Translating (1) into QL₁ as ‘\( \forall x (Gx \land Hx) \)’ would plainly be wrong. For that says, quite differently, that everyone is a wise logician!

(ii) Translating (2) as ‘\( \exists x (Gx \rightarrow Hx) \)’ would also be wrong. For take some non-logicician; Socrates will do. Then (Socrates is a logician \( \rightarrow \) Socrates is wise) is
§23.1 Restricted quantifiers again 221

true, being a material conditional with a false antecedent. Hence there is some
x such that (x is a logician → x is wise). Hence ‘∃x(Gx ⊃ Hx)’ is true. But
obviously the existence of a non-logician like Socrates isn’t enough to make
it true that some logicians are wise.

Further, we want the translations of intuitively equivalent ordinary-language propositions
to come out as formally equivalent. Translating restricted ‘all’ with a conditional and
restricted ‘some’ with a conjunction makes this happen. For example:

(iii) ‘Not all logicians are unwise’ is equivalent to ‘Some logicians are wise’. So
their QL₁ translations should be equivalent.

Translating the restricted ‘all’ with a conditional, the first gets rendered
¬∀x(Gx ⊃ ¬Hx), which is equivalent to ‘∃x¬(Gx ⊃ ¬Hx)’ (why?). But we
know from propositional logic that something of the form ¬(α ⊃ β) is
equivalent to (α ∧ ¬β); and similarly, just because of the meaning of the
connectives, ‘∃x¬(Gx ⊃ ¬Hx)’ is equivalent to ‘∃x(Gx ∧ Hx)’.

So putting things together, ‘¬∀x(Gx ⊃ ¬Hx)’ comes out as equivalent to
‘∃x(Gx ∧ Hx)’, which translates ‘Some logicians are wise’. As required.

(b) We are essentially taking it that

(1) All logicians are wise
comes to the same as

(1’) Anyone x is such that, if x is a logician, x is wise.

What kind of ‘if’ is involved here? In rendering (1) and equivalently (1’) into QL₁ as

(3) ∀x(Gx → Hx)

we are assuming that the ‘if’ in (1’) works like a material conditional. But that seems
correct, whatever is the case with other ‘if’s. Why?

(1’) will be false if any instance ‘if n is a logician, n is wise’ is false. But we can’t
make (1’) false by looking at facts about non-logicians. So for any non-logician n, the
instance ‘if n is a logician, n is wise’ can’t be false. In other words, the conditional ‘if
n is a logician, n is wise’ has to be true when the antecedent is false. Which is exactly
what the truth-functional reading of the conditional gives us. (That’s brisk – but in fact
we have already made the same point in §16.5, so we need not labour over it again.)

(c) The English plural in (2) ‘Some logicians are wise’ surely indicates that there is
more than one wise logician. But (4) ‘∃x(Gx ∧ Hx)’, our official translation into QL₁,
says only that there is at least one. Is this minor discrepancy worth fussing about?

Almost never. When we propose something of the form ‘Some Gs are H’ as a premiss,
or aim to derive it as a conclusion, the number of Gs that are H typically doesn’t matter
to us – we care about whether any of them are. So we can simply swallow the small
translational infelicity in rendering ‘Some Gs are H’ as ‘∃x(Gx ∧ Hx)’. In almost every
case it is worth the gain in formal simplicity. And in any case, when we later introduce
an augmented language which we will call QL₂, we will see how to express ‘some (more
than one)’ with the formal resources of our then enriched language, should we really
need to do so.
23.2 Existential import

(a) Here’s another issue about our translation of restricted quantifiers. We have just claimed that, setting worries about plurals aside, propositions of the form

(1) All Gs are H,
(2) Some Gs are H,

can be rendered into a QL language by corresponding wffs of the form

(3) $\forall x(Gx \rightarrow Hx),$
(4) $\exists x(Gx \land Hx).$

But now note that, traditionally, it has been supposed that a universally quantified proposition like (1) has ‘existential import’ – for instance, if it is true that all logicians are rational, then there must exist some logicians to be wise. Hence (1) entails (2), or so the story goes.

However, the wff (3) doesn’t entail (4) (assuming it is possible for there to be no Gs). For suppose that there are no Gs in the relevant domain. Then trivially (4) is false. But everyone x is such that (x is G $\rightarrow$ x is H), because each instance (n is G $\rightarrow$ n is H) is true, because the antecedent of the material conditional is always false. So (3) is then true. Hence, it is possible for (3) to be true and (4) false. So (3) doesn’t entail (4).

To be sure, (3) plus the existential assumption ‘$\exists xGx$’ will entail (4). But the point is that this additional existential assumption is needed.

So the moral is this: if the traditional view is right, then a proposition of the form (1) entails (2) even though (3) by itself doesn’t entail (4) – hence our translations aren’t adequate because they don’t respect logical relations.

(b) We might well wonder, however, whether tradition gets it right. Consider for example Newton’s Second Law in the form

(5) All particles subject to no net force have constant velocity.

Surely to accept Newton’s Law is not to commit ourselves to the actual existence of any such particles: doesn’t the law remain true even if, as it happens, every particle is subject to some net force? Another example: couldn’t the notice

(6) All trespassers will be prosecuted

be true even if there are no trespassers; indeed it could well be because (6) is true that there are no trespassers!

(However, to complicate matters, it might be suggested that there is a subtle difference here between ‘all’ and ‘any’ – with All Gs are H being more apt for cases where there are some Gs, while Any Gs are H leaves it open whether there are some Gs – so perhaps we should really state Newton’s law with ‘any’?)

(c) We fortunately need not get entangled in such debates. We need not settle whether vernacular propositions of the form All Gs are H usually entail the existence of Gs. Just regard this as one of those messy issues about ordinary language which get helpfully tidied away when we go formal.

Our policy henceforth will be to take the default rendering of ‘all’ propositions to be along the lines of ‘$\forall x(Gx \rightarrow Hx)$’, which doesn’t entail the existence of Gs. We can then
always tack an explicit existential clause ‘\(\exists xGx\)’ onto the translation if, on a particular occasion, we think it matters.

(d) Continuing to suppose that ‘\(G\)’ expresses the property of being \(G\), we should note that both ‘\(\forall x(Gx \to Hx)\)’ and ‘\(\forall x(Gx \to \neg Hx)\)’ will be true if there are no \(G\)s (why?). But we can live with that. After all, a common way of proving that there are no \(G\)s is to show that if there were any such things, they would have to be \(H\) but also have to be not-\(H\) too, which is impossible. Or in formal terms, we establish that both ‘\(\forall x(Gx \to Hx)\)’ and ‘\(\forall x(Gx \to \neg Hx)\)’ are true and deduce ‘\(\forall x \neg Gx\)’.

23.3 ‘No’

(a) Our QL languages have just two flavours of built-in quantifiers: should we add more? In particular, what about adding a ‘no’ quantifier?

Well, note that in QL\(_1\), ‘\(\forall x \neg Hx\)’ means that everyone is not wise – in other words, no one is wise. In the same language ‘\(\neg \exists x Hx\)’ means that it isn’t the case there there is someone who is wise. So this too means that no one is wise. Hence we already have two very simple QL\(_1\) translations of a basic ‘no’ proposition (and they are equivalent because ‘\(\forall x \neg Hx\)’ is equivalent to ‘\(\neg \exists x \neg \neg Hx\)’ and we can lose the double negation).

The point obviously generalizes. Putting it informally, we can recast ‘no’ propositions using a quantifier prefix ‘(nothing \(x\) is such that)’, and then we can render that prefix by either of the corresponding formal expressions ‘\(\forall x \neg\)’ or ‘\(\neg \exists x\)’. Or putting that more carefully, with the same notation as in §22.3,

| Take a QL language whose quantifiers range over the \(D\)s. Suppose we interpret the name \(\nu\) as denoting \(n\), and suppose the wff \(\alpha(\nu)\) then says that \(n\) is thus-and-so. Then \(\forall \xi \neg \alpha(\xi)\) and \(\neg \exists \xi \alpha(\xi)\) are equivalent, and say that there is nothing \(x\) (among the \(D\)s) is such that \(x\) is thus-and-so – i.e. no \(D\) is thus-and-so. |

Of course, we could have added a third built-in quantifier symbol to QL languages (e.g. a rotated ‘\(N\)’ to go with ‘\(V\)’ and ‘\(E\)’) so we can express ‘there are no \(F\)s’ by something like ‘\(NxFx\)’. This would make the language a bit more complicated and, more annoyingly, it would increase the number of rules needed later for dealing with quantifier arguments. Yet the gain would be minor, since we can easily enough express the ordinary-language ‘no’ quantifier using negation plus one of the quantifiers we already have.

(b) And what about restricted ‘no’ quantifications? For example, how can we render into QL\(_1\) the calumny that

(1) No philosopher is wise?

As with other ‘no’ translations, we have two options, one using a universal quantifier, one using an existential quantifier. The first option formalizes the thought that everyone is such that, if a philosopher, then not wise:

(2) ‘\(\forall x(Fx \to \neg Hx)\)’.

The second option formalizes the equivalent thought that there doesn’t exist someone who is a philosopher and wise:
Why are these equivalent? Because $\forall x (Fx \rightarrow \neg Hx)$ is equivalent to $\neg \exists x (Fx \rightarrow \neg Hx)$, and that in turn is equivalent to $\neg \exists x (Fx \land Hx)$ – for just recall that $\neg (\alpha \rightarrow \neg \beta)$ is equivalent to $(\alpha \land \beta)$.

23.4 Translations into QL₂

Having explained how to restrict quantifiers and how to deal with ‘no’ propositions, let’s put these lessons to work in translating a raft of propositions into a QL language. We will use some frankly rather silly examples. But nothing would be gained by taking sober mathematical examples (for instance), and we could run the risk of making things look more difficult than they really are.

Let’s therefore define the following language:

In QL₂, the proper names are just
- $m$: Maldwyn,
- $n$: Nerys,
- $o$: Owen;

and the predicates are just
- $F$: ① is a man,
- $G$: ① is a woman,
- $L$: ① loves ②,
- $M$: ① is married to ②,
- $R$: ① is a child of ② and ③,

(where the arities of the formal predicates match that of their interpretations).

The quantifiers of QL₂ range over all living people.

And now consider how to translate the following bunch of propositions into QL₂ (we are interested in how to translate propositions, not whether they are true!):

1. There’s someone who loves Maldwyn, Nerys and Owen.
2. Whoever is loved by Owen is loved by Maldwyn too.
3. Every woman who loves Maldwyn is loved by Owen.
4. Maldwyn loves some woman who loves Owen.
5. Nerys and Owen love any child of theirs.
6. Owen is Maldwyn’s child.
7. It isn’t true that Maldwyn loves everyone
8. Maldwyn is married to no one.
9. No one who loves Nerys loves Owen.
10. If everyone loves Nerys then Owen does.
11. If anyone loves Nerys then Owen does.
12. If anyone is loved by Maldwyn then they are loved by Owen too.
13. Every man loves someone or other.
23.4 Translations into QL$_2$

(15) Someone Owen loves is loved by everyone Nerys loves.
(16) No woman loves every man.
(17) No woman loves any man.
(18) Anyone who is married loves someone they aren’t married to.
(19) A married man only loves women.
(20) No one married loves anyone who loves Nerys.
(21) Not every married man loves any woman who loves him.

You might suspect that one or two of these ordinary language claims are ambiguous. But all the more reason, then, for wanting to use a regimented, unambiguous, formalized language in order to explore the logic of disambiguated claims.

(b) We will translate in stages, following the plan explained in §22.3. So we go from the original English to a version using generic prefixed quantifiers plus variables. Then we abbreviate the quantifier prefixes using our quantifier symbols, replace ordinary language connectives with the standard formal connectives, giving us a Loglish intermediate step. Then we use the glossary for QL$_2$ to render the names and predicates.

Reaching an initial version with prefixed generic quantifiers itself typically requires two steps. We will first use prefixed restricted quantifiers – either simple ones such as ‘some man $x$ is such that’, ‘every woman $y$ is such that’, or else slightly more complex ones involving relative clauses like ‘everyone $x$ who is married is such that’ or ‘some woman $y$ who loves Owen is such that’. Then we render these restricted quantifiers using generic quantifiers over the whole domain which are restricted by conditionals and conjunctions.

(1) There’s someone who loves Maldwyn, Nerys and Owen

\[ \exists x (Lxm \land (Lxn \land Lxo)). \]

The internal bracketing of the three-ply conjunction is up to you. And of course, there is nothing significant about the choice of informal variable ‘$x$’ or the formal variable ‘$x$’. As in the rest of our examples too, you can use any variables you like in your translations, so long as you preserve the crucial pattern of linkages from quantifier prefixes to slots in later expressions.
Whoever is loved by Owen is loved by Maldwyn too
\[ \therefore (\text{Everyone } x \text{ who is loved by Owen is such that} \text{ Maldwyn loves } x) \]
\[ \therefore (\text{Everyone } x \text{ is such that} (\text{if Owen loves } x \text{ then Maldwyn loves } x)) \]
\[ \forall x(\text{LOx} \rightarrow \text{Lmx}). \]

Every woman who loves Maldwyn is loved by Owen
\[ \therefore (\text{Every woman } x \text{ who loves Maldwyn is such that} \text{ Owen loves } x) \]
\[ \therefore (\text{Everyone } x \text{ is such that} (x \text{ is a woman who loves Maldwyn, then} \text{ Owen loves } x)) \]
\[ \forall x((\text{Gx} \land \text{Lmx}) \rightarrow \text{LOx}). \]

Maldwyn loves some woman who loves Owen
\[ \therefore (\text{Some woman } x \text{ who loves Owen is such that} \text{ Maldwyn loves } x) \]
\[ \therefore (\text{Someone } x \text{ is such that} (x \text{ is a woman who loves Owen} \land \text{ Maldwyn loves } x)) \]
\[ \exists x((\text{Gx} \land \text{Lxo}) \land \text{Lmx}). \]

Nerys and Owen love any child of theirs
\[ \therefore (\text{Everyone } x \text{ who is a child of Nerys and Owen is such that} \text{ Nerys loves } x \land \text{ Owen loves } x) \]
\[ \therefore (\text{Everyone } x \text{ is such that} (x \text{ is a child of Nerys and Owen} \rightarrow \text{ Nerys loves } x \land \text{ Owen loves } x)) \]
\[ \forall x(\text{Rxno} \rightarrow (\text{Lnx} \land \text{Lox})). \]

Owen is Maldwyn’s child
\[ \therefore (\text{Someone } x \text{ is such that} \text{ Owen is the child of Maldwyn and } x) \]
\[ \exists x \text{Romx}. \]

The point of this last little example is to emphasize that \( \text{QL}_2 \)'s predicate ‘\( R \)' is a ternary predicate, glossed as ‘\( 0 \) is a child of \( 2 \) and \( 3 \)’. If we want to translate the ordinary-language binary ‘child of’ predicate, then we have to render the English by a formal version of ‘\( 0 \) is a child of \( 2 \) and someone’ (or equivalently, of course, ‘\( 0 \) is a child of someone and \( 2 \)’). We quite often have to use this sort of trick when constrained to translate into a given \( \text{QL} \) language – we will see a similar example in a moment.

From now on, we start using square brackets informally to mark off significant parts of a complex ordinary-language or Loglish proposition. For example, we have:

It isn’t true that Maldwyn loves everyone
\[ \therefore \neg [\text{Maldwyn loves everyone}] \]
\[ \therefore \neg [(\text{Everyone } x \text{ is such that} \text{ Maldwyn loves } x)] \]
\[ \neg \forall x \text{Lmx}. \]

Here, the square brackets helpfully highlight the scope of the initial negation. We can then, as it were, work inside the square brackets – so we take the first square-bracketed expression and rephrase it to give us the second square-bracketed expression.

Our next two examples involve the ordinary language ‘no’ quantifier. The previous
§23.4 Translations into QL₂

section tells us that that ‘no’ propositions can be rendered in two different ways, with ‘some’ and negation, or with ‘every’ and negation. So:

(8) Maldwin is married to no one
   ≈ It is not the case that [Maldwin is married to someone]
   ≈ ¬∃xMmx.

Or, starting again, taking the alternative route:
   ≈ (Everyone x is such that) Maldwin isn’t married to x
   ≈ ∀x¬Mmx.

(9) No one who loves Nerys loves Owen
   ≈ It is not the case that [someone who loves Nerys loves Owen]
   ≈ ¬∃x[(there is someone x who loves Nerys and is such that) x loves Owen]
   ≈ ¬∃x[(there is someone x who is such that)(x loves Nerys and x loves Owen)]
   ≈ ¬∃x(Lxn ∧ Lxo).

Or alternatively
   ≈ Everyone who loves Nerys does not love Owen
   ≈ (Everyone x who loves Nerys is such that) x doesn’t love Owen
   ≈ (Everyone x is such that)(x loves Nerys → x doesn’t love Owen)
   ≈ ∀x(Lxn → ¬Lxo).

The next proposition is a conditional with complete propositions as antecedent and consequent. So we bracket its translation as follows:

(10) If everyone loves Nerys then Owen does
   ≈ If [everyone loves Nerys], then [Owen loves Nerys]
   ≈ [(everyone x is such that) x loves Nerys] → [Owen loves Nerys]
   ≈ (∀xLxn → Lon).

By contrast, in (11) on its natural reading, the scopes of the conditional and the quantifier are reversed (we noted before, in §20.1, that ‘every’ and ‘any’ can behave differently in the antecedents of conditionals):

(11) If anyone loves Nerys then Owen does
   ≈ Anyone is such that, if they love Nerys, then Owen loves Nerys
   ≈ ∀x(Lxn → Lon).

There is an alternative, arguably equally natural, translation of (11), which picks up the point that in some contexts ‘any’ can be replaced by ‘some’, as here:

If anyone loves Nerys then Owen does
   ≈ If [someone loves Nerys] then [Owen loves Nerys]
   ≈ (∃xLxn → Lon).

Later, we will formally show that these alternative translations are indeed equivalent, as we need.

Note, though, that we can’t always translate ‘any’ in the antecedent of a conditional with an existential quantifier. For consider
(12) If anyone is loved by Maldwyn then they are loved by Owen too.
   \[ \forall x (Lmx \rightarrow Lox). \]

In this case, we can’t just put an existential quantifier inside the conditional, as that would leave the variable in the consequent dangling (think about it?).

(d) Pausing for breath, we now move on to the examples with multiple quantifiers. Again, we will informally use square brackets to highlight parts of complex expressions, and then we work within the square brackets as we move from one stage to the next:

(13) Every man loves someone or other
   \[ \exists x (Fx \land \exists y (Lxy)). \]

An alternative translation in this case would be ‘\( \forall x \exists y (Fx \rightarrow Lxy) \)’. But careful! – we can’t always move quantifiers from inside to outside brackets like this (see §??).

(14) Some man loves some woman
   \[ \exists x (Fx \land \exists y (Gy \land Lxy)). \]

An alternative translation would be ‘\( \exists x \exists y ((Fx \land Gy) \land Lxy) \)’.

(15) Someone Owen loves is loved by everyone Nerys loves
   \[ \exists x (Lox \land \forall y (Lny \rightarrow Lxy)). \]

(16) No woman loves every man
   \[ \neg \exists x (Gx \land \forall y (Fy \rightarrow Lxy)). \]
§23.4 Translations into QL₂

Or, starting again, taking the alternative route for translating a ‘no’ proposition:

\[ \text{Every woman is such that she doesn’t love every man} \]
\[ \Leftrightarrow (\text{Everyone } x \text{ is such that } (\text{if } x \text{ is a woman, then } \neg [(\text{everyone } y \text{ is such that if } y \text{ is a man, } x \text{ loves } y])] \]
\[ \Leftrightarrow \forall y (Gx \to \neg \forall y (Fy \to Lxy)). \]

The stressed claim ‘No woman loves just any man’ is naturally heard as saying the same as (16). But an unstressed (17) ‘No woman loves any man’ is more naturally read as we might read ‘No woman loves some man’ (compare ‘No five year old reads any volumes of Proust’). Taking this reading, and recycling our translation of (14), we have

\[ \text{No woman loves any man} \]
\[ \Leftrightarrow \text{It is not the case that } [\text{some woman loves some man}] \]
\[ \Leftrightarrow \neg \exists x (Gx \land \exists y (Fy \land Lxy)). \]

An equally good rendition, using our other style for translating ‘no’ propositions, is
\[ \forall x (Gx \to \neg \exists y (Fy \land Lxy)). \]

Here’s a third:

\[ \text{No woman loves any man} \]
\[ \Leftrightarrow \text{Every woman is such that every man is such that she doesn’t love him} \]
\[ \Leftrightarrow (\text{Everyone } x \text{ is such that } [(\text{someone } y \text{ is such that }) \text{ if } x \text{ is a man, then } x \text{ doesn’t love } y]) \]
\[ \Leftrightarrow \forall x (Gx \to \forall y (Fy \to \neg Lxy)). \]

Can you see why these three renditions indeed ought to be equivalent to each other?

Our next example involves translating the unary predicate ‘is married’ into a language which only has a built-in binary predicate meaning ‘is married to’. No problem! – we just rely on the obvious semantic equivalence of ‘is married’ to ‘is married to someone’. Thus we have:

\[ \text{Anyone who is married loves someone they aren’t married to} \]
\[ \Leftrightarrow (\text{Everyone } x \text{ who is married is such that there is someone } y \text{ whom } x \text{ isn’t married to, such that } x \text{ loves } y] \]
\[ \Leftrightarrow (\text{Everyone } x \text{ is such that } [(\text{someone } y \text{ is such that } x \text{ isn’t married to } y \text{ and } x \text{ loves } y])] \]
\[ \Leftrightarrow \forall x (x \text{ is married } \to \exists y (\neg Mxy \land Lxy)). \]

Finally, we render ‘x is married’ using ‘M’ plus an existential quantifier – and the only question is what variable to use. A good rule of thumb is this: when you need to introduce some variable into a wff, always use one that doesn’t already appear in the wff. That way you will avoid unintended tangles. So, we can finish
\[ \forall x (\exists z Mxz \to \exists y (\neg Mxy \land Lxy)). \]

(In §7?, when we talk about QL syntax again, we’ll return to this example, and we will then see why it would also have been permissible to re-use ‘y’ in the antecedent of the
conditional and write ‘$\exists y Mxy$’ there. But that’s a story for later, and we won’t pause over this now.)

In the next example, ‘A married man . . . ’ is naturally read as a universal generalization about married men. So, we can begin

(19) A married man only loves women
≈ (Every man $x$ who is married is such that)($x$ loves only women)
≈ (Everyone $x$ is such that)(if $x$ is a man and is married, then [$x$ loves only women])

Now, ‘$x$ only loves women’ is in turn naturally read as saying anyone whom $x$ loves is a woman (but we will want to leave it open as to whether there is anyone $x$ in fact loves). So, continuing,

≈ (Everyone $x$ is such that)($(x$ is a man and is married) → [(everyone $y$ is such that) if $x$ loves then $y$ is a woman])
≈ $\forall x((Fx \land \exists z Mxz) \to \forall y(Lxy \to Gy))$.

Our next example involves that troublesome vernacular quantifier ‘anyone’ again. As have already seen, in some contexts, ‘anyone’ is replaceable by ‘someone’; (20) seems to be another case in point. The following rendition seems natural:

(20) No one married loves anyone who loves Nerys.
≈ (Everyone $x$ who is married is such that) it is not the case that [there is someone who loves Nerys whom $x$ loves]
≈ (Everyone $x$ who is married is such that) $\neg$[(there is someone $y$ such that) $y$ loves Nerys $\land x$ loves $y$]
≈ (Everyone $x$ is such that) $x$ is married $\to$ $\neg$[(there is someone $y$ such that) $y$ loves Nerys $\land x$ loves $y$]
≈ $\forall x(\exists z Mxz \to \neg \exists y(Lyn \land Lxy))$.

An alternative translation could be ‘$\forall x(\exists z Mxz \rightarrow (\forall y)(Lyn \rightarrow \neg Lxy))$’. Why are these equivalent?

(21) Not every married man loves any woman who loves him
≈ It is not the case that (Every man $x$ who is married is such that)($x$ loves any woman who loves $x$)
≈ $\neg$(Everyone $x$ is such that)(if $x$ is a man and is married, then [$x$ loves any woman who loves $x$])
≈ $\neg$(Everyone $x$ is such that)(if $x$ is a man and is married, then [(every woman $y$ is such that) if $y$ loves $x$ then $x$ loves $y$])
≈ $\neg$(Everyone $x$ is such that)(if $x$ is a man and is married, then [(everyone $y$ is such that) if $y$ is a woman, then if $y$ loves $x$ then $x$ loves $y$])
≈ $\neg \forall x((Fx \land \exists z Mxz) \to \forall y((Gy \land Lyx) \to Lxy))$.

A close alternative would be ‘$\neg \forall x((Fx \land \exists z Mxz) \to \forall y((Gy \land Lyx) \to Lxy))$’.

And those, surely, are enough examples of English-to-QL translation to illustrate the basic stage-by-stage approach! There are more examples to try in this chapter’s Exercises, and many more illustrations too in the following chapters.
23.5 Translations from QL₂

Translating from an unfamiliar language tends to be a lot easier than translating into that language. Once we have learnt to spot the devices which QL languages use for expressing restricted quantifications and ‘no’ quantifiers, it is often pretty easy to construe a wff straight off.

And if the worst comes to the worst, and we are faced with a more dauntingly complex wff, we can always simply reverse the step-by-step procedure that we have just been using. To illustrate this, we will take three QL₂ wffs:

(1) \( \neg \forall x \forall y ((Fx \land Gy) \rightarrow (Mxy \rightarrow \exists zRzxy)) \)
(2) \( \forall x(Gx \rightarrow \forall y \forall z (Rxyz \rightarrow (Lxy \land Lxz))) \)
(3) \( (\forall x(Gx \rightarrow \neg Lxm) \rightarrow \forall x((Lxm \rightarrow \exists yRxy) \land (\exists yRxy \rightarrow Lxm))) \)

We now unpack these wffs in stages to arrive at interpretations, for example as follows:

(1) \( \neg \forall x \forall y ((Fx \land Gy) \rightarrow (Mxy \rightarrow \exists zRzxy)) \)
  \( \equiv \neg (\text{For any } x \text{ and } y \text{ a woman, then if } x \text{ is a man and } y \text{ a woman, there is someone } z \text{ who is } x \text{ and } y \text{’s child}) \)
  \( \equiv \neg (\text{For any man and any woman, if they are married to each other, they have a child together}) \)
  \( \equiv \text{Not every man and woman who are married have a child together.} \)

(2) \( \forall x(Gx \rightarrow \forall y \forall z (Rxyz \rightarrow (Lxy \land Lxz))) \)
  \( \equiv (\text{For anyone } x \text{ if } x \text{ is a woman then [for anyone } y \text{ and anyone } z \text{ if } x \text{ is a child of } y \text{ and } z, then } x \text{ loves } y \text{ and } x \text{ loves } z] \)
  \( \equiv \text{Any woman } x \text{ is such that [for anyone } y \text{ and anyone } z \text{ if } x \text{ is a child of } y \text{ and } z \text{ then } x \text{ loves } y \text{ and } x \text{ loves } z] \)
  \( \equiv \text{Any woman loves her parents.} \)

(3) \( (\forall x(Gx \rightarrow \neg Lxm) \rightarrow \forall x((Lxm \rightarrow \exists yRxy) \land (\exists yRxy \rightarrow Lxm))) \)
  \( \equiv \text{If } \forall x(Gx \rightarrow \neg Lxm), \text{ then } \forall x((Lxm \rightarrow \exists yRxy) \land (\exists yRxy \rightarrow Lxm)) \)
  \( \equiv \text{If no woman loves Maldwyn, then } \forall x(Lxm \text{ if and only if } \exists yRxy) \)
  \( \equiv \text{If no woman loves Maldwyn then, for any } x, x \text{ loves Maldwyn if and only if } x \text{ is a child of Maldwyn and someone} \)
  \( \equiv \text{If no woman loves Maldwyn, then he is loved just by his children, if any!} \)

Again, you will meet plenty more examples in exercises and later chapters.

23.6 Choosing the domain

We finish this action-packed chapter by discussing two more general issues.

We have been talking about translating into a given QL language. But it is worth asking: how do we choose a language in the first place, if we are in the business of regimenting some argument(s), prior to evaluating them? In particular, how do we fix on an appropriate domain of quantification to use?

Suppose we want to formalize the whole argument

(1) Some philosophers are logicians; all logicians are wise; so some philosophers are wise.
Then of course, the obvious thing to do is to choose a formal language like $\mathbf{QL}_1$, with an inclusive domain containing enough people to include all the philosophers and logicians. And then we can express the restricted quantifications in the premisses and conclusion like this:

$$(2) \exists x (Fx \land Gx), \forall x (Gx \rightarrow Hx) \rightarrow \exists x (Fx \land Hx).$$

For another example, suppose we are mathematicians wanting to formalize the similar argument

$$(3) \text{Some numbers are perfect numbers; all perfect numbers are composite numbers.}$$

This time, the natural thing to do might well be to take a language whose domain of quantification is already restricted to just numbers (in particular, natural numbers, i.e. integers from zero up), and then – in that language – we can express the argument along the following lines:

$$(4) \exists x P x, \forall x (P x \rightarrow C x) \rightarrow \exists x C x.$$ 

Note: since the current domain contains just numbers, we don’t need to further restrict our formal quantifiers to express the first premiss and the conclusion.

Alternatively, however, if our interest is less in the arithmetic of natural numbers and more in the logic, and we want a way to bring out the similarity of (1) and (3), we might choose a somewhat different formal language, perhaps with a wider domain, and with a predicate ‘$N$’ for ‘is a number’. Then we can render (3) by

$$(5) \exists x (Nx \land P x), \forall x (P x \rightarrow C x) \rightarrow \exists x (Nx \land C x),$$

which is formally just like (2), choice of predicate letters apart.

These simple examples illustrate an important moral. Suppose we are given some ordinary language propositions with explicitly restricted quantifiers like ‘some philosophers’, ‘some numbers’, . . . . Then whether the corresponding formal quantifiers need to be explicitly restricted or not will depend on the domain of the $\mathbf{QL}$ language we choose to use: is it more or less inclusive? And which language we choose to use will depend on context, e.g. on what else we want to be able to translate at the same time, or on what parallels with other arguments we want to bring out, etc. In formalizing, we bring out as much structure as we need to do for the purposes in hand. (We will revisit this point in §23.8.)

23.7 ‘Translating’ again

(a) Early in the last century, we find Bertrand Russell and Ludwig Wittgenstein, much influenced by Frege, being gripped by the thought that the surface look of ordinary language disguises the true ‘logical form’ of propositions. They proposed that a central concern of philosopher-logicians should be to ‘analyse’ propositions to reveal this putative underlying structure. And the idea later gained ground that the now standard notation of the new quantifier/variable logic perspicuously represents real structures of ‘logical form’, hidden by the surface syntax of language. Is there anything in this idea?

Here’s an obvious problem. As we have seen, if we try to render e.g. ‘No woman loves Maldwin’ into a language like $\mathbf{QL}_2$, we arrive at one of ‘$\neg \exists x (G x \land L x m)$’ or
‘∀x(Gx → ¬Lxm)’. And it really just doesn’t seem very plausible to suggest that the first formal wff here is picking up on an existential quantifier and a conjunction somehow already hidden under the surface of the ordinary English. Or should we prefer the second rendition and say instead that there is a hidden universal quantifier and a conditional? How could we possibly choose?

Note, however, that our rendition of the ‘no’ proposition into $QL_2$ involves working around two difficulties that we have voluntarily imposed on ourselves. To keep our formal language simple, we have stipulated that there isn’t a ‘no’ quantifier built into the language, and also that the remaining quantifiers range over the whole domain (in this case, over everyone) with restrictions to be made by using connectives. So the particular shapes of our formal versions of ‘No woman loves Maldwin’ are really an artefact of these two choices we made in designing our formal $QL$ languages; it would be hard to argue that these should be deeply revealing about ordinary language claims.

Still, what about the most fundamental thing, the basic quantifier/variable idea for representing general claims? Frege, Russell and Wittgenstein are now surely onto something.

Everyday English doesn’t vividly mark in surface syntax the deep semantic difference between a proper name like ‘Maldwyn’ or ‘Nerys’ on the one hand, and an expression of generality like ‘someone’ or ‘nobody’ on the other hand. Grammatically speaking, such expressions typically behave in much the same way – e.g. we can plug either kind into predicates like ‘is Welsh’ or ‘loves’ and get grammatical sentences. Now, it would be a step too far to say that expression of generality behave exactly like proper names, and can always grammatically occupy the same positions in sentences. For example, contrast ‘Something wicked this way comes’ with the ungrammatical ‘Jack wicked this way comes’, or ‘Someone brave rescued the dog’ with ‘Jill brave rescued the dog’. Or contrast ‘Foolish Donald tweeted’ with the ungrammatical ‘Foolish nobody tweeted’. And so on. But still, the key point remains that the surface syntax of everyday language closely assimilates names and (some) expressions of generality, and doesn’t explicitly and systematically signal the semantic divide between names and quantifiers in the way that $QL$ syntax does. For example, everyday language does not mark the key fact that proper names are scopeless while quantifiers have scopes, and hence can generate scope-ambiguities – while, by contrast, our formal notation sharply distinguishes a name like ‘m’ from a quantifier expression like ‘∃x…x’ (with the prefixed quantifier and its linked variables overtly indicating the scope of the generality).

So, yes, it might well be said that the quantifier/variable notation represents more perspicuously aspects of semantic structure which are somewhat masked by the surface grammar of English.

(b) But we need not really press that last point.

For our key claim is that it is a good strategy for logical purposes to avoid the messy complexities of English usage – think again of ‘any’ vs ‘every’! – by (1) translating arguments involving general claims into a nicely behaved, unambiguous, tidy $QL$ language, and then (2) evaluating the arguments in their tidily regimented form. For this strategy to work, it is not necessary to suppose that the artificial languages (even in part) reflect structures already there under the surface of English. It is enough that wffs in the
formal language behave in truth-relevant ways which sufficiently respect the contents of the claims made in the original English.

Now, whenever we deploy this divide-and-rule strategy for assessing arguments – i.e. whenever we translate from ordinary language into a formal language, then deal with the clean-and-tidy formalized arguments – we would ideally like both (i) very direct, natural, unforced translations into the relevant artificial language L, and (ii) an elegantly simple language L which is easy to manipulate and theorize about. But the more ‘natural’ we try to make the translations between English and L, the greater the number of fine distinctions we may need to mark in L, and so the more complex everything will get. And that will mean, in particular, increased complexity when we come to evaluate arguments in L.

In practice, then, we are often faced with a trade-off between closeness of translation and ease of logical manipulation. We’ve seen this before when we adopted the material conditional, giving us the simplicity of truth-functional logic for PLC, but at the price of some rather dubious-seeming translations of everyday ‘if’ s. And again, in the present case, there’s a trade-off. We buy the relative simplicity of QL languages at the cost of, among other things, having to shoehorn our everyday restricted quantifications into a language where such quantifications have to be mocked up using conjunctions and conditionals. But the price is right: the formal translations, although sometimes a little cumbersome, enable us to capture the truth-relevant content of ordinary claims well enough. So we can use our formal translations in assessing the validity of a great number of arguments relying on the presence of quantifiers. And by those standards, the use of QL languages is a pretty resounding success.

Or so, let’s hope, the following chapters will reveal.

23.8 Summary

We need not add a ‘no’ quantifier to our QL languages, given that we can so easily render something of the form ‘There are no Fs’ by the corresponding wff ‘¬∃xFx’ or equivalently ‘∀x¬Fx’.

Translating from English into a QL language can be thought of as taking four main stages:

1. Rephrase a given English proposition using prefixed-quantifiers-plus-variables. Typically, we will need to use restricted quantifiers, and then render the restricted quantifiers by using generic quantifiers over the whole domain restricted by conditionals and conjunctions.

2. Replace vernacular pronouns with variables, and make cross-linkings clear by tagging quantifier prefixes with the variables they are respectively linked to.

3. Use ‘∀’ and ‘∃’ to abbreviate quantifier prefixes, replace vernacular connectives with formal one, giving us Loglish expressions.

4. Then use the glossary for the relevant QL language to render the names and predicates.
§23.8 Summary

Restricted quantifiers will get translated along the following lines:

- All Fs are G ≡ ∀x(Fx → Gx),
- Some Fs are G ≡ ∃x(Fx ∧ Gx),
- No Fs are G ≡ ∀x(Fx → ¬Gx) or ¬∃x(Fx ∧ Gx).

We not claim that our renditions of e.g. restricted quantifiers reveal the underlying logical form of our vernacular propositions – so that ‘All Fs are G’ is unmasked as really a quantified conditional (though the formal quantifier/variable representation of general claims arguably is revealing about the semantic nature of ordinary quantifiers too). Our key claim is only that our formal translations track the (tidied-up, disambiguated) truth-relevant content of vernacular propositions well enough for the purposes of our divide-and-rule approach to logic.