Notes

QL proofs

- The second edition of my *Introduction to Formal Logic* will have chapters on natural deduction in the main text (and supplementary chapters on truth trees, pretty much as in the first edition, are available online). So here are revised draft chapters on QL languages followed by new draft Chapters 31–34 on natural deduction for predicate logic.
- There will exercises added to the end of these chapters, and worked answers online. I'm not one of those who fetishizes doing a lot of exercises, teaching proof strategies etc. Basic understanding is what we should be aiming for, say I!
- I have included a long-winded version of the Table of Contents at the outset, so you can – if you want – see how these chapters follow on from earlier chapters in the overall structure of the book.
- Since Chapters 31–34 are new, all comments/corrections will be most gratefully received!

General notes to readers

- What follows is an excerpt from a planned second edition of my *Introduction to Formal Logic* (CUP, 2003). This is a textbook aimed at first-year philosophy students: if all goes well, the second edition will appear in 2019 in the *Cambridge Introductions to Philosophy* series. The initial Chapters 1 to 7 introduce some key concepts in a relaxed way, and should be useful background whatever formal logic course you then go on to take. Then, from Chapter 8, we make a start on propositional logic. The Table of Contents will tell you what these chapters cover.
- Work still needs to be done on the exercise sets at the end of the chapters. There will eventually be answers to the exercises on the web.
- At this stage, all kinds of comments (other than ones that mean, in effect, 'You are writing the wrong book!') are most welcome. Obviously I want to hear about any typos you spot. But in addition, it is very useful to hear e.g. about passages that you found obscure or more difficult than others, and passages you thought were possibly

misleading. And if you are not a native English-speaker, do note any words or turns of phrase that you found puzzling.

- Spelling follows British English rather than North American English conventions. So I write e.g. 'fulfil' rather than 'fulfill', 'skilful' rather than 'skillful', etc. I aim to systematically use 'ize' endings where appropriate. (But still, if you think what I've written is a typo, do say so – I'd rather you over-corrected than under-corrected!)
- One issue about punctuation. Suppose we have some introductory words followed by some indented displayed material. Should the introductory words end with a colon or not? My general line is this: if the introduction is a complete clause, I use a colon; if the thought runs on straight into the indented material, I don't.
- I use 'they'/'them' as gender-neutral singular pronouns, except when the resulting English strikes me as rebarbative, in which case I paraphrase away. (Let me know of cases that really grate.)
- Unresolved references of the kind '**??**' are forward-looking to chapters or sections that aren't included in the current chapters and will be resolved in due course.
- Comments and corrections please to ps218 at cam dot ac dot uk – if you do send any, it could be *very* helpful if you mention the date of this draft.

An Introduction to

Formal Logic

Second edition

Peter Smith

June 10, 2019

Contents

 Comments to ps218@cam.ac.uk

Interlude: Formalizing general propositions

(a) We have now approached questions of validity for PL inferences in two rather different ways.

(1) In Chapter 15 we defined the notion of tautological validity, and we then explored the brute-force truth-table method for determining validity in this sense.

(2) Then, since the last Interlude, we have developed a natural deduction framework for showing the validity of inferences depending on the core sentential connectives, using intuitively correct step-by-step derivations.

There are various ways of doing (2), but we have opted for a Fitch-style layout for our PL proof system (we eventually codified this in §24.2, and we need not summarize it again here). Happily, we can show – although we relegated details to an Appendix – that the two approaches validate just the same inferences. Our proof system doesn't overshoot by allowing us to construct proofs which aren't truth-preserving (the system is sound). And it doesn't undershoot by failing to validate some correct reasoning with the connectives (the system is complete).

And now, after all this work on propositional logic, we must move on. So where next?

(b) Consider the following simple arguments:

(1) Felix is a cat. All cats are scary. So, Felix is scary.
(2) Ludwig envies anyone Alma loves. Alma loves Fritz. So, Ludwig envies Fritz.
(3) Each great philosopher is worth reading. Some logicians are also great philosophers. So some logicians are worth reading.
(4) Some philosophers admire every logician. No philosopher admires any charlatan. So, no logician is a charlatan.

These are logically valid. But if we expose no more structure than the way in which whole propositional clauses are related by connectives (if any), then each of these arguments can only be assigned the utterly unreliable inferential form *A, B, so C*.

To explain the validity of these arguments, we will therefore have to dig deeper inside their premisses and conclusions. What is crucial here is the presence of expressions of generality – i.e. *quantifiers* such as 'all', 'any', 'every', 'each', not to mention 'some' and 'no'. If we are to make any headway tackling the logic of general statements, we are therefore going to need a way of handling sub-propositional structure, and of explaining in particular the roles of quantifiers.

(c) Ordinary language has a rich multiplicity of quantifiers. For example, 'all', 'any', 'every', and 'each' can *all* be used to express a general claim about everything of some kind. Though, as we will soon see, these four quantifiers also behave in significantly different ways. And this is just one of the complications we encounter with vernacular quantifiers. But we surely don't want to get bogged down in all the linguistic niceties.

So we will follow the 'divide and rule' strategy we have already used to deal with propositional arguments. Just as we sidestepped the quirks of ordinary language by going formal in order to deal with the propositional connectives, the plan will be to introduce some well-behaved formal languages for regimenting ordinary-language quantified arguments, and then to investigate the logic of the neatly formalized arguments.

However, formalization in this case is rather easier said than done! The design brief for PL languages as explained in §8.7 is very simple and the resulting languages are easy to work with. By contrast, the design rationale for our formal languages for quantifier logic is less immediately obvious. And the resulting QL languages look a lot less like ordinary language. So we need to spend some time explaining carefully why these new languages have the form they do. And it will require some effort to become fluent at handling them.

(d) We go slowly. Take the 'Felix' argument again. As we said, this argument's validity depends on the internal sub-propositional structures of its premisses and conclusion. And apart from the quantifier 'all', it is also crucial how the name 'Felix' recurs, and the way that '(is a) cat' and '(is/are) scary' recur.

When we formalize this sort of argument in a QL language we will therefore need some way of representing the internal structure of the relevant propositions by using formal *names* and *predicates*. So, to guide the syntax and semantics of our formalization, we need some preliminary remarks about proper names and logical predicates in ordinary language, and about the simple sentences built from them. That's the business of the next chapter, Chapter 25.

Following this, in Chapters 26 and 27, we motivate and explain the general idea of a *quantifier-variable notation* for expressing general propositions in a disciplined way.

Then in Chapter 28 we learn how to implement this pivotal Fregean idea in the formal setting of QL languages.

And that won't be the end of the story (for example, we will later want to add *function expressions* to our QL languages). But for the next few chapters, we concentrate on the basics.

25 Names and predicates

As just indicated in the Interlude, before turning to consider the proper treatment of quantifiers in our formal languages, we need to start with some preliminary remarks about names and predicates.

25.1 Names, and other 'terms'

(a) A language like English (here we include mathematical English) has a variety of devices for referring to particular objects. These include:

(i) *Proper names* – e.g. 'Alice', 'Socrates', 'Kilimanjaro', 'Vienna', '0', 'π', etc. (Ordinary proper names can of course be widely shared, as anyone called 'Peter Smith' can tell you. So context will be needed to fix *which* Peter Smith is being referred to by a particular use of the name.)

(ii) *Demonstratives* or demonstratively used *pronouns* – e.g. 'this', 'that (guy)', or simply 'she' (relying on context to fix the intended reference).

(iii) So-called *definite descriptions* of the form 'the F' – e.g. 'the tallest girl in the class' or 'the hardest logic problem' (once again, some context might be needed in these cases to fix the reference: which class, which set of problems?).

(iv) *Functional terms* – where we use an expression for a function like 'the mother of', 'the population of', 'the positive square root of', in combination with some other referring expression, to form a complex term like 'the mother of Alice', 'the mother of the mother of Alice', 'the father of that guy', 'the population of this town', 'the positive square root of π', '$\pi/2$', '$\sin(\pi/2)$', ...

Rather differently, we also use what are variously called temporary names, quasi-names, arbitrary names, or ...

(v) *Dummy names* – e.g. 'John Doe' and 'Jane Roe' in the lawyer's 'Suppose Jane Roe is the executor of John Doe's will, ...'; or consider the mathematician's use of 'a' in 'Let the constant a be an arbitrary real number, ...'.

(b) In contrast to ordinary language, the logical languages in this book will not have *any* referring expressions which rely on context for their reference. But we will add expressions of type (i), (v) and then (iv) in stages, while setting aside definite descriptions for special treatment. So:

(1) Initially, our QL languages will only have the equivalent of unstructured *proper names* which unambiguously always refer to the same fixed thing.

(2) Next, we will add *dummy names* to our formal languages, as these will play a key role in regimenting arguments with quantifiers.
(3) Eventually, we will also be able to form *functional terms* in our QL languages.

There is a standard label that generically covers expressions of these three types:

> Proper names, dummy names and functional terms are all *terms*.

More specifically, these are *singular terms* as opposed to plural terms (like 'the Muses', 'the wives of Henry VIII', 'the fifth roots of i' which refer to more than one thing), and also as opposed to general terms for kinds of thing (like 'man' or 'book' or 'electron').

We concentrate first on those singular terms which are proper names, and for both informal and formal cases we say (unsurprisingly!):

> The object referred to by a proper name is the name's *reference*.

25.2 Predicates and their 'arity'

(a) According to school-room grammar, simple sentences can typically be parsed into 'subject' and 'predicate' (roughly, the subject tells us what/who the sentence is about, the predicate then tells us something concerning that subject-matter). For example, the following sentences are traditionally divided into subject then predicate as shown:

(1) Socrates / is wise
(2) Socrates / loves Plato
(3) Some philosopher / is wise

But modern logicians see things differently. First, we will insist that there is a deep difference between the roles of names like 'Socrates' and of quantifier-involving expressions like 'some philosopher', and so resist classifying them together as 'subjects'.

Second – the point emphasized in this section – we will generalize the school-room notion of a predicate. For us, a predicate is an expression which can combine with one *or more* singular terms or other expressions to form a sentence. For example, 'loves' in 'Socrates loves Plato' counts as a predicate in the logician's sense: it is a binary predicate taking two suitable expressions to form a sentence.

(b) It is convenient to have a device to use when we want to mark explicitly *where* one or more names can be attached to a predicate to form a sentence (let's for now take simple combinations with names to be the most basic case). We can use dots and dashes, as in '... is a cat', '... loves —'. But here's another standard device: we can use numbered counters to mark slots where names can go, as in '① is a cat', '① loves ②', '① is between ② and ③', etc.

To be clear: earlier, we informally used an expression like 'n is a man' to represent a *whole* sentence, with the schematic letter standing in for some particular name (see e.g. §3.3). By contrast, '① is a man' represents just *part* of a sentence, a predicate, with the counter indicating where a name (or other expression) would need to be added to get a sentence.

So: the ordinary-language predicates '① is a cat', '① is scary', '① mews', '① killed themself' (for example) take one name or other suitable completion to form a sentence. While '① loves ②', '① is a pupil of ②', '① took ② to task', '① and ② are married to each other' all take two names, different or the same, to form a sentence. And similarly, '① is between ② and ③' takes three names. Call these unary, binary, ternary predicates respectively (or informally, one-place, two-place, three-place predicates). And generalizing:

> A k-ary predicate is an expression which takes k names (or other suitable expressions) to form a sentence.
>
> The number of names that a predicate takes to form a sentence is its *arity*.

(c) Consider, however, the English sentences

(1) Tom and Dick cohabit,
(2) Tom, Dick and Harry cohabit,
(3) Tom, Dick, Harry and Jo cohabit,

and so on. Or for a logical example, consider

(1′) 'P' and 'Q' entail '(P ∧ Q)',
(2′) 'P', 'Q' and 'R' entail '((P ∧ Q) ∧ R)',
(3′) 'P', 'Q', 'R' and 'S' entail '(((P ∧ Q) ∧ R) ∧ S)',

and so on. Do such examples show that some ordinary-language predicates like 'cohabit' (or 'entail') can combine with *varying* numbers of singular terms for people (or terms for wffs) to form a sentence? Arguably so. However, when we go formal, we will insist that our built-in QL predicates are tidily behaved and each has a definite arity; so in particular, each combines with a *fixed* number of terms to make an atomic wff.

(d) Now to refine our notation and the associated notion of arity. There is a useful convention that places attached to predicates which are marked in the same way are to be filled in the same way – for example, we can fill out '① killed ①' to get 'Seneca killed Seneca' and 'Mark Anthony killed Mark Anthony' but not 'Nero killed Poppaea'. So we can say that '① killed ①' takes *one* name, albeit twice over; and it is naturally classed as another *unary* predicate because it is equivalent to '① killed themself'.

Understood this way, the arity of a predicate is represented by the number of *different* counters that need to be attached. (We do, however, allow different slot-marking counters to be filled out the same way – compare §3.2(b). So 'Seneca killed Seneca' counts as a way of filling out the binary predicate '① killed ②' alongside 'Nero killed Poppaea'.)

25.3 Predicates, properties and relations

(a) What then do predicates *do*, semantically? Informally: unary predicates express properties, binary predicates express relations between two things (or between a thing and itself), ternary predicates express three-way relations, and so on.

For example, in the sentence 'Socrates is snub-nosed', the unary predicate '① is snub-nosed' is used to ascribe the property of being snub-nosed to Socrates. In the sentence

'Romeo loves Juliet', the binary predicate '① loves ②' is used to say that the binary relation of loving holds between Romeo and Juliet taken in that order, with the first loving the second. Life and love being what they are, it matters which name goes into which slot attached to the predicate here: for Romeo can stand in the relation of loving to Juliet without, conversely, Juliet loving Romeo. Similarly, in the sentence 'Stevenage is on the rail line between Cambridge and London', the predicate '① is on the rail line between ② and ③' expresses a ternary relation between the three towns – again, the order in which their names appear matters.

(b) So far, so good. Except that the notion of a property/relation does come freighted with a *lot* of unwanted baggage. For example, many philosophers urge us to distinguish what they take to be genuine properties like *being human* and *being wise* and *having zero mass* from trumped-up fakes like *being either a logician or a neutrino or the Eiffel Tower* and *being green if it is Tuesday and tasting like coffee otherwise*. And philosophers argue a lot about where to draw the genuine/fake line.

By contrast, logicians are generous souls who talk about properties and relations in a cheerfully inclusive way (for we certainly don't want to get entangled with the philosophers' metaphysical debates). For us, then, talk about predicates expressing properties/relations will officially mean no more than this:

> A unary predicate *expresses a condition* that an object o has to meet if the predicate is to be true of o; a binary predicate expresses a condition that the objects o_1, o_2 taken in that order have to meet if the predicate is to be true of them; and so on – where any condition, however complex or gerrymandered, is countenanced.

(c) Now for some useful terminology:

> If an object o meets the condition expressed by a given unary predicate, then – for short – we will say that o *satisfies* the predicate.
>
> Likewise, suppose that the two objects o_1 and o_2, taken in that order, meet the condition expressed by a binary predicate, then we will say that (o_1, o_2), i.e. the ordered pair of o_1 and o_2, satisfies the predicate.

After ordered pairs come ordered triples, quadruples, quintuples, etc. There is nothing mysterious about finite ordered sequences of objects like these. The traditional general term for them is *tuples*. So a k-tuple is just a finite ordered sequence of k objects $(o_1, o_2, \ldots, o_k)$ – and we can take a 1-tuple to be an object by itself. Then:

> If a k-tuple of objects meets the condition expressed a k-ary predicate, then the k-tuple *satisfies* the predicate.

25.4 Predicates: sense vs extension

(a) We said in §7.2 that the sense of a declarative sentence is the aspect or ingredient of its meaning that fixes the condition under which the sentence is true. Similarly, the *sense of a unary predicate* – to start with that case – *is the aspect of its meaning which*

fixes the condition which some object must meet if the predicate is to be true of it. So, to borrow a Fregean example, compare the predicates '(is a) horse/steed/nag/gee-gee'. The use of these alternative expressions will no doubt give a different colouring to the sentences which contain them. However, these predicates arguably share the same sense, i.e. they can be taken to mean the same in truth-relevant respects and express the same property. In other words, they express the same equine condition that an object has to meet if the predicate is to be true of it.

(b) The sense of a sentence is, to repeat, the truth-relevant aspect of its meaning. But knowing the sense of a sentence isn't, in general, enough to tell us whether the sentence actually *is* true; that usually depends on how things are in the world. And it is exactly similar with predicates. The sense of a unary predicate is the truth-relevant aspect of its meaning. But knowing the sense of a predicate – i.e. knowing the condition it expresses – isn't, in general, enough to tell us which objects *do* satisfy the predicate. That usually depends on how things are in the world. Let's say that

> The objects which *do* satisfy a unary predicate form the predicate's *extension*.

Then the point we have just made is that the extension of a predicate will in general depend on the situation in the world.

Take for example the unary predicate 'is a philosopher'. In the world as it is, Socrates is in the predicate's extension, along with Plato and Aristotle. But he might not have been. For example, there is another way the world might have been in which Socrates opts for a quiet life and doesn't hang around the agora corrupting the youth. And perhaps Socrates in that possible world has an argumentative sister (who doesn't exist in the actual world), and *she* satisfies 'philosopher' there.

(c) Evidently, then, a predicate with a fixed sense can acquire different extensions as we consider different possible ways the world might be. Conversely, predicates with different senses can have the same actual extension.

For example, the two sentences 'Jo is a human' and 'Jo is a terrestial featherless biped' express different thoughts, because the predicates 'is a human' and 'is a terrestial featherless biped' express different conditions. Something could in theory satisfy one condition without satisfying the other: in some possible worlds, there are humans who are not terrestial (humans could move to Mars), and there are terrestial featherless bipeds who aren't human (suppose Neanderthals survive). But still, these two predicates happen to have the same extension as things stand – or so our traditional example assumes.

For another example, the predicates 'is a unicorn' and 'is a dragon' plainly do not have the same sense – satisfying the condition for being a unicorn is not the same thing as satisfying the condition for being a dragon, and you could have a world with creatures of the one kind but not the other. That's why to ask whether dragons ever roamed the earth is not to ask whether unicorns ever roamed the earth. But these two predicates in fact happen to have the same empty or null extension in the world as it is.

(d) The sense of a binary relational predicate is likewise that aspect of its meaning which fixes the condition which an ordered pair of objects must meet in order to satisfy

the predicate. And pairs which satisfy the condition will form the predicate's extension. Generalizing, the sense of a k-ary predicate fixes the condition on which k-tuples of objects satisfy the predicate. And

> The k-tuples which satisfy a k-ary predicate form the predicate's extension.

Again, k-ary predicates with different senses can have the same extension. While, keeping its sense fixed, a k-ary predicate can have different extensions in different possible situations.

(e) Note, we have just talked about some objects (or ordered pairs of objects, etc.) as 'forming' the extension of a predicate, and talked too about those objects as being 'in' the extension of a predicate. It very natural, then, to speak of extensions as being *sets* of objects (or *sets* of ordered pairs of objects, etc.).

We will fall in with this convenient way of speaking. But in doing this, we can leave it open just what we are committing ourselves to. Do we need to think of the extension of '① is a philosopher' as a new object in its own right, over and above the people who satisfy the predicate? Or can we take talk of a set of philosophers here just as a *façon de parler* – a handy way of referring in the singular to the philosophers, plural? We can remain neutral. (But for more on the issues, see the Appendix 'On sets'.)

25.5 Names: sense vs reference

(a) It is pretty much agreed on all sides that, as just explained, we need a sense vs extension distinction for predicates. Do we need an analogous sense vs reference distinction for proper names? Frege famously thought so, but the issue is hotly contended.

Here, just to think about, is an argument for the view that we *do* need some sense vs reference distinction for names.

> For vividness, we can take an engaging fictional example. 'Superman' and 'Clark Kent' are two different names for the same person. It is evidently possible for someone to have both names in their vocabulary but not to know that these *are* two names for the same person. Lois Lane is initially in this state. And these names are then associated by her with very different Fregean 'modes of determination' – i.e. very different ways of picking out what is, as it happens, the same person. So, roughly speaking, she thinks of Superman as the handsome superhero in tights, and of Clark Kent as the bespectacled reporter. *That* is why Lois can rationally assert 'Superman can fly' yet deny 'Clark Kent can fly' even though, in the circumstances, these sentences attribute the same property to the same person and therefore must in fact have the same truth value. And that's also how 'Superman *is* Clark Kent' can be quite startlingly informative for her.
>
> Hence, for the purposes of interpreting Lois's talk about 'Superman' and 'Clark', and in order to make her related behaviour comprehensible, it seems that we need to know more than merely the reference of the name she uses. We do need to distinguish the different 'modes of determination' which Lois associates with the different names – or, as Frege would put it, we need to distinguish the different

senses the names have for her. Which is why, just as two predicates with the same extension can have different senses, we need to recognize that two names with the same reference can have different senses.

Or so, in the very briefest terms, a Fregean argument begins. However, all this is contentious, and we certainly can't follow the twists and turns of the debates here. Fortunately, for our purposes we don't need to. Why so?

25.6 Reference, extension, and truth

(a) When is a basic name/predicate sentence true? When the named object meets the condition expressed by the predicate. Or equivalently, given our definitions:

> A sentence formed from a name and a unary predicate is true if and only if the reference of the name is in the predicate's extension.

The sense of a predicate combines with how things are in the world to fix the extension of the predicate: but it is this extension which then does the work of determining the truth values of basic sentences involving that predicate. Similarly, whether or not names have senses, it is the reference of a name which determines the truth values of basic name/predicate sentences involving that name. And generalizing,

> A sentence built from a k-ary predicate and k names in order is true if and only if the k-tuple of objects referred to by the names is in the extension of the predicate.

For example, the extension of the predicate '① loves ②' (defined over people) comprises those ordered pairs of people such that the first of them loves the second. And a sentence of the form 'm loves n' is true just when the pair of objects denoted by 'm' and 'n', in that order, is in the extension of the predicate.

What goes for these basic sentences also goes for other sentences which are simple enough. In other words, as far as the constituent names and predicates are concerned, the truth values of such sentences are fixed by the references of names and extensions of predicates.

(b) Why, though, the restriction to 'simple enough' sentences? Because ordinary language does allow more complex contexts where e.g. the extension of a predicate *isn't* what matters for truth. So compare the claims

(1) Daisy believes that Dobbin is a unicorn.
(2) Daisy believes that Dobbin is a dragon.

Little Daisy may indeed believe that Dobbin, dressed up for the fête, is a real unicorn without believing him to be a dragon! Which shows that (1) and (2) can have *different* truth values even though the predicates '① is a unicorn' and '① is a dragon' have the *same* null extension.

How can this be so? Plainly, what makes (1) true and (2) false are facts about the contents of Daisy's beliefs, facts about the way she is thinking. And a Fregean will say that these contents are given by the respective *senses* of 'Dobbin is a unicorn' and 'Dobbin is a dragon' and hence are determined by the *senses* of the predicates here.

So, in this case, it isn't the shared null extension of the predicates which contributes to fixing the truth values of (1) and (2) but rather the predicates' different senses.

Next, compare

(3) Lois believes that Superman can fly.
(4) Lois believes that Clark Kent can fly.

Still going along with the familiar fiction, initially (3) is true while (4) is false. How can this be, given the names have the same reference?

A Fregean will say that (3) and (4) make claims about the contents of Lois's beliefs, and these contents are given by the senses of 'Superman can fly' and 'Clark Kent can fly', and hence are determined by the senses of 'Superman' and 'Clark Kent'. In these cases, the shared reference of the names isn't what determines truth values.

(c) We need not worry whether the Fregean is right about these cases. Because, in contrast with ordinary language, our QL languages *won't* allow us to embed names and predicates inside a context like '*n* believes that . . . '. In the relevant way, QL *sentences will all be simple enough for their truth values to be fixed by references and extensions alone.*

But if senses don't have to enter the story about what fixes the truth values of QL sentences, they also won't have to enter into the story about QL validity (since validity is defined in terms of truth preservation). So – as far as our logical enquiry into QL arguments goes – we won't need to have a developed theory of sense. We should be grateful for small mercies!

25.7 Summary

As we will see, the non-logical building blocks of QL languages – at least at the initial stage – are names and predicates (only later will we add function expressions). This chapter prepares the ground for moving to these formalized languages by saying something about names and predicates in ordinary language.

We distinguish proper names from other ways of picking out particular objects (unlike definite descriptions and functional terms, they are not internally complex and, unlike demonstratives, names can be treated as context-independent).

A logical predicate is an expression that takes, in particular, some names to form a sentence, and which expresses some condition which the named objects are to meet if the sentence is to be true.

We need to distinguish the sense of a predicate from its extension. Arguably, we need a parallel distinction between the sense and reference of a name.

It is the references of names and the extensions of predicates that determine the truth values of simple sentences built from names and logical predicates, both in ordinary language and in QL languages. But in QL languages, as we will see, this applies to all the complex sentences too – what matters about names and predicates in fixing the truth values of sentences will still be references and extensions.

26 Quantifiers in ordinary language

So much, then, for some introductory ideas about names and predicates. We now turn to consider how we are going to express general propositions in our formal languages.

Expressions of generality in a natural language like English behave in complex ways: this chapter explores just some of the tangles. The following chapter then explains a strategy for handling quantification in formalized languages so as to avoid these many vagaries of ordinary language by using a so-called quantifier-variable notation.

26.1 Which quantifiers?

Aristotle in his *Prior Analytics* concentrates on forms of quantified proposition corresponding to the English 'All F are G', 'No F are G' , 'Some F are G', and 'Some F are not G'. It is, by the way, worth knowing that propositions of these types later came to be called, respectively, A, E, I and O propositions – supposedly, the vowels are from the Latin *affirmo* (I affirm, for the positive two) and *nego* (I deny, for the negative two).

But that's just four among many ordinary-language constructions involving expressions which quantify. Consider, for example:

> Every/any/each F is G,
> At most one/at least three/exactly five F are G,
> Finitely/infinitely many F are G,
> There are exactly as many F as G,
> Few/many/almost all F are G,
> Half/a quarter of the F are G,
> Numerous/a lot of/several/enough F are G.

And arguably, as we'll see in Chapter **??**,

> The F is G

belongs on the list too.

Those constructions all take two general terms (replacing 'F' and 'G') to form a sentence. By contrast, consider

> Everything/something/nothing is G,
> Everyone/everybody/someone/somebody/no one/nobody is G,
> There is/there exists a G,
> There exist two/at least four/many Gs,
> There are infinitely many Gs.

In these cases we have to replace just the one schematic letter '*G*' with a general term to get a sentence.

We can naturally speak, then, of *binary* vs *unary* quantifier constructions. And a first question is: which of these varied binary and unary constructions are we going to care about in developing our logical quantification theory?

We will in fact be focusing on so-called *universal* ('every'/'any'/'all'/'each') and *existential* ('some'/'there is') quantifiers. These both come in binary and unary versions: we'll consider in the next chapter how the two versions are related. We will also get a treatment of 'no' quantifiers for free because we will have negation in our formal languages and 'no' is equivalent to 'not some'. Later, when we add an identity predicate to our formal languages, we will be able to cope with numerical quantifiers like 'at least five/exactly five/at most five' and we can also give a formal treatment of 'the'. This way, we arguably get all the quantifiers we need for core mathematical and scientific purposes, and for a great amount of common-or-garden reasoning too.

26.2 Every/any/all/each

(a) Consider the following English sentences, used as stand-alone claims:

(1) Every play by Shakespeare is worth seeing.
(2) Any play by Shakespeare is worth seeing.

These surely come to the just same, expressing the same generalization about all the plays. Yet compare how (1) and (2) embed in wider contexts. For example, constrast

(3) If every play by Shakespeare is worth seeing, I'll eat my hat.
(4) If any play by Shakespeare is worth seeing, I'll eat my hat.

Some find (4) ambiguous. But on the natural reading – expressing a general dismissal of the Bard's plays – (4) says something quite different from (3). Similarly for the following:

(5) I don't believe that every play by Shakespeare is worth seeing.
(6) I don't believe that any play by Shakespeare is worth seeing.

On the natural reading of (6), it says something quite different from (5).

So the quantifiers 'every' and 'any' are certainly not *always* interchangeable. For another example to show this, consider the following pair (a 'Monro' is a Scottish mountain over 3000 feet):

(7) If Jack can climb every Monro, he can climb every Monro.
(8) If Jack can climb any Monro, he can climb every Monro.

The first is a tautology; the second could well be false on the reading which says that, if Jack can climb any one Monro, then he can climb them all, even the most challenging.

(b) What about 'all' and 'each'? Take for example

(9) All plays by Shakespeare are worth seeing.
(10) Each play by Shakespeare is worth seeing.

As stand-alone claims, these again seem to convey just the same message as (1) and (2). And now consider

(11) If all plays by Shakespeare are worth seeing, I'll eat my hat.
(12) I don't believe that all plays by Shakespeare are worth seeing.

(11) is equivalent to (3), and (12) to (5). So, at least in these cases, sentences containing 'all' behave like sentences with 'every' rather than 'any'. Similarly for the corresponding 'each' sentences.

So can we perhaps say this: 'all', 'each' and 'every' are just stylistic alternatives, the main difference being that 'all' goes with a plural construction while 'each' and 'every' go with the corresponding singular form? Well, we will also have to allow some other grammatical adjustments (for example, 'All my books are blue' corresponds to 'Each of my books is blue' and 'Every one of my books is blue'). But now compare

(13) You can fit all (of) Jane Austen's novels into a single, ordinary-sized, book.
(14) You can fit each of Jane Austen's novels into a single, ordinary-sized, book.

I take the first to be false, the second true; 'all' here is naturally read as 'all, taken together', 'each' is more naturally read as 'each, taken separately'. Similarly, compare

(15) Jill can legally marry all (of) Bill's brothers.
(16) Jill can legally marry each of Bill's brothers.

By my lights, the second may be true, while the first isn't given our laws against polygamy.

But if you disagree (perhaps you find the sentences in each of the last pairs to be ambiguous between the taken-together/taken-separately readings), then no matter: your view equally reinforces the point that these English expressions of generality do behave in complex ways. And we haven't yet mentioned other complications – for example, the way that 'all' but not 'every/each' can go with so-called mass terms: e.g. we can say 'All snow is white' but not 'Every/each snow is white'.

(c) Given that we are interested in the logic of arguments involving ideas of generality, but are not especially interested in theorizing about e.g. the tangled English usage of 'every/any/all/each', what to do?

As already announced in the last Interlude, we will follow the same strategy that we introduced to cope with the connectives. We will explain how to construct QL languages which avoid the complex multiplicity of the English 'every/any/all/each' quantifiers by having just *one*, simply-behaved, way of forming universal generalizations of this type. These formalized languages will also replace 'some' and its close relations 'there is at least one' and 'there exists' with a single simply-behaved substitute. We can now divide and rule again (see §8.6); in other words, we can tackle the assessment of many quantificational inferences in two stages. We translate a given argument into a suitable QL language, enabling us to represent quantified messages in a uniform, tidy, and easily understood way. And *then* we assess the resulting formalized argument for validity.

26.3 Quantifiers and scope

Wanting to side-step the tricky multiplicities of English usage is one major motivation for going more formal. However, there is another, deeper, reason for adopting an artificial language to regiment quantified propositions for logical purposes. For we need to avoid the structural ambiguities which beset English quantified sentences. Let's explore.

(a) For a warm-up exercise, compare the following sentences:

(1) Some senior Republican senators took cocaine in the nineteen-nineties.
(2) Some women died from illegal abortions in the nineteen-nineties.

As a shock headline, (1) can be read as saying, of some present Republican senators, that they have a dark secret in their past (namely that they dabbled in illegal drugs in the nineties). But (2) is not to be read, ludicrously, as saying of some present women that they have a secret in *their* past (namely that they earlier died of an illegal abortion). The intended claim is of course that some women living in the nineties died then from illegal abortions.

With some mangling of tenses, and adding brackets for maximal clarity, we might represent the intended readings of the two sentences respectively as follows:

(3) (Some senior Republican senators are such that)(in the nineteen-nineties it was true that) they use cocaine.
(4) (In the nineteen-nineties it was true that)(some women are such that) they die from illegal abortions.

Here the final verbs are now deemed to be tenseless. And so a compelling way of describing the situation is to say that the tense modifier and the quantification are understood as having different *scopes* in the messages intended by (1) and (2) – compare §8.5 on the idea of scope. But note that the shared surface form of those two original sentences doesn't explicitly mark this difference. We implicitly rely on context and plausibility considerations in order to arrive at the natural readings (3) and (4).

Now consider

(5) Some philosophers used to be enthusiastic logical positivists.

Are we saying that some current philosophers went through a positivist phase in their brash younger days? Or are we saying that, once upon a time, some then philosophers were keen positivists? The intended message could be

(6) (Some philosophers are such that)(it used to be true that) they are enthusiastic logical positivists,

or alternatively we could be claiming

(7) (It used to be true that)(some philosophers are such that) they are enthusiastic logical positivists.

Without any context to help us construe (5), the claim is simply ambiguous. And the ambiguity is structural (the individual words aren't ambiguous). The issue is: which has wider scope, i.e. which governs more of the sentence, the generalizing operator or the tense modifier?

(b) Mixing together ordinary expressions of generality and tense-modifying operators is therefore prone to produce scope ambiguities. So too is mixing vernacular quantifiers with what logicians call *modal* operators – i.e. with expressions of necessity and possibility.

Consider, for example, that notorious philosophical claim

(8) Every perceptual experience is possibly delusory.

Does this mean that each and every perceptual experience is one which, taken separately, is open to doubt, i.e.

(9) (Every perceptual experience is such that)(it is possible that) it is delusory?

Or is the thought that the whole lot could be delusory all at once, i.e.

(10) (It is possible that)(every perceptual experience is such that) it is delusory?

These claims are certainly different, and it is philosophically important which reading is meant e.g. in arguments for scepticism about the senses. Even if (9) is true, (10) doesn't follow. Compare, for example,

(11) Every competitor might win first prize,

and the following two readings:

(12) (Every competitor is such that)(it is possible that) they win first prize,
(13) (It is possible that)(every competitor is such that) they win first prize.

It could be that (12) is true because it's a fairly run competition of skill with very well matched entrants, while (13) is false because the rules governing the knock-out rounds ensure that only one ultimate winner can emerge.

Now, some English-speakers rather firmly claim that (8) means (9); others equally firmly say it means (10). I take that as some evidence for a third view, namely that in ordinary use (8) is dangerously ambiguous. In what order is the intended message built up using the quantifier and the modality? Surface grammar doesn't fix this. (Again, you don't have to agree. Even if you think that, in correct usage, (8) and (11) are unambiguous, you must still acknowledge the key point, namely that the interpretation of such quantified sentences crucially involves paying attention to questions of scope.)

(c) Exploring the logic of tense operators and of modal operators is beyond the scope(!) of this book. So, let's turn to the sorts of cases that more immediately concern us. First, consider what happens when we mix everyday expressions of generality with *negation*.

Here is one kind of example. A statement of the form 'Some *F*s are not *G*s' is usually heard as being entirely consistent with the corresponding 'Some *F*s are *G*s'. For example, 'Some students are not good at logic' is no doubt sadly true; but it is consistent with the happier truth 'Some students are good at logic'.

But now suppose Jack says 'Some footballers deserve to earn five million a year'. Jill, in response, takes a high moral tone, expostulates about the evils of huge wages in a world of poverty and deprivation, and emphatically concludes: 'So certainly, some footballers do *not* deserve to earn five million a year'. Jill plainly does not intend to be heard as saying something compatible with Jack's original remark! Which shows that, although vernacular English sentences of the form 'Some *F*s are not *G*s' are more usually construed as meaning

(14) (Some *F*s are such that)(it is not the case that) they are *G*s,

in some contexts the negation can be understood to have wide scope, with the resulting message being

(15) (It is not the case that)(some *F*s are such that) they are *G*s.

Another example. In the *Merchant of Venice*, Portia says

(16) All that glisters is not gold,

which is naturally construed in context as meaning that not everything that glisters (i.e. glitters) is gold. So what Portia intends to say is equivalent to

(17) (It is not the case that)(all that glisters is such that) it is gold.

But now compare the following sentence which in surface form is just like (16):

(18) All that perishes is not divine.

This is more naturally read with the quantifier and the negation scoped the other way around, i.e. as equivalent to

(19) (All that perishes is such that)(it is not the case that) it is divine.

In sum: like the logical operation of negation, quantifying may be thought of as an operation which can govern more or less of a complex sentence. And when an assertion mixes a negation and a quantifier, we may not be able to read off the intended relative scopes of the two logical operators simply from the surface form of the sentence being used – unless we set out to be very ploddingly explicit in the manner of e.g. (17) and (19). We can again be left with a scope ambiguity – as in these examples, perhaps:

(20) Jill didn't finish every book.
(21) Everyone's not yet arrived.
(22) I would not give that to anyone.

(d) Consider next some sentences containing more than one quantifier:

(23) On Mother's Day, every mother will get some gift.
(24) At the Crack of Doom, the clouds will part and every human will look up to see some angry and jealous god.

The first is surely to be read as saying that every mother will get her own present; the second is more naturally read as saying that some angry and jealous god will reveal itself to all alive that dread day. We can express the core parts of these claims respectively as follows:

(25) (Every mother is such that)(there is some gift such that) she will get it.
(26) (There is some angry and jealous god such that)(every human is such that) they will look up to see it.

So on the obvious reading of (23), the 'every' quantifier has the wider scope, i.e. governs more of the sentence, while we read (24) with the 'some' quantifier having wider scope.

Here is another example of an 'every/some' sentence, this time one which can quite reasonably be read with the quantifiers scoped either way:

(27) Everyone loves a certain someone.

Is the claim that each person (perhaps in some contextually indicated group) has their own beloved? Or is the claim that there is someone who is the apple of every eye?

Given an English sentence involving more than one quantifier, context and plausibility considerations may indeed often serve to privilege one particular reading and prevent misunderstanding – as is the case with (23) and (24). But there is always the potential for unresolved ambiguity, as perhaps in (27).

(e) Note, by the way, that these four kinds of scope ambiguity, arising when everyday quantifiers are mixed with tense, modality and negation and with other quantifiers, do not occur when *proper names* are used instead of the quantifiers. Consider, for example:

(28) Jack took cocaine in the nineteen-nineties.
(29) Jill might possibly win.
(30) Michael does not deserve five million a year.
(31) Socrates loves Plato.

We need to fix who is being talked about and the unit of currency. But none of these claims is *structurally* ambiguous – issues of scope do not arise. Unlike quantifiers, genuine names are always *scopeless*.

(f) Finally, having introduced the notion of quantifier scope, let's revisit a couple of examples from the previous section:

(32) If every play by Shakespeare is worth seeing, I'll eat my hat.
(33) If any play by Shakespeare is worth seeing, I'll eat my hat.

The difference between these can now be seen as involving issues of scope. In (32), the generalization is definitely confined inside the antecedent of the conditional – if this narrow-scope generalization is true, I'll eat my hat. By constrast, the more natural reading of (33) gives the generalization wider scope, outside the conditional – any play by Shakespeare is such that, if that one is worth seeing, I'll eat my hat.

There might be some mileage in the view that (in the absence of other indications) 'every' defaults to narrow scope with respect to other logical operators, and 'any' defaults to wide scope. But everyday usage really doesn't seem to be consistent in this respect: I continue to find (8) scope-ambiguous, for example.

(g) We have said enough. Yes, we can argue about the interpretation of particular examples: but, there is no question that ordinary language expressions of generality can generate scope ambiguities. Part of the design brief for our formal QL languages, therefore, will be that – as with PL languages – they must be constructed so that the scope of logical operators, now including the quantifiers, is always clear and unambiguous.

26.4 Fixing domains

Suppose that, at the beginning of class, I ask 'Is everyone here?'. Plainly I am not asking e.g. about everyone now living. I have in mind those who have signed up for the logic class, and I am asking if all of *them* are present. I continue: 'I am impressed: someone solved that extremely tricky optional homework problem!'. Again, I am not talking about some student in another college or in another year, let alone just some random person; I mean that there is someone among the current logic class who solved the problem. So, although not explicitly restricted, 'everyone' and 'someone' here are naturally understood as ranging over a rather particular group of people; these people form the intended current *domain* of the quantifiers – or, in rather grander terminology, they make up the *universe of discourse*. (An 'every/any/all/each' quantifier is universal in the sense of making a claim about every object in the relevant universe of discourse.)

As in our logic class example, we often leave it up to the context to fix the domain which the quantifiers are ranging over, i.e. to determine who or what currently counts as 'everyone' or 'everything'. For more examples, consider e.g. 'Everyone enjoyed the concert', 'Make sure everyone has a glass of wine', 'Everything fitted into two suitcases', 'Can you finish everything this morning?'.

Even if we use a binary quantifier with a general term qualifying the quantifier, what we are quantifying over can still vary with context: consider e.g. 'Every seat is booked', 'All passengers have now boarded', 'Any laptops should now be switched off', 'Each baby started crying' – which seats, which passengers, which laptops, which babies?

Moreover, we are adept at following mid-conversation *shifts* in the intended domain of our quantifiers, using contextual cues and common sense to follow implicit changes in who counts as 'everyone' or what counts as 'everything' (or who/what counts as a relevant 'someone'/'something'). To continue with our logic class example, suppose I go on to say: 'As we saw, everyone agrees that modus ponens is a good form of inference for the ordinary-language conditional'. You immediately pick up that I am this time generalizing not over the members of the class but over e.g. the mainstream logical authors whom we were explicitly discussing last week. And when I later remark, 'Someone, however, has denied that modus ponens is always reliable', you don't take me to be contradicting myself: you charitably suppose that I am now moving on to cast the generalizing net rather wider, beyond the standard authors we've already mentioned, to include logical outliers and mavericks.

Ordinary discourse, then, often leaves it merely implicit who or what we are quantifying over. But when we start regimenting arguments for logical purposes – with the aim of making everything ideally clear and determinate – we won't want to rely on contextual clues or interpretative charity. We will need a policy for explicitly and clearly fixing the domains of quantifiers in QL languages.

26.5 Summary

Ordinary English expressions of generality include both unary and binary quantifiers. These behave in complicatedly tangled ways. Consider e.g. the availability of 'all', 'every', 'any', and 'each' to express universal generalizations, and compare the different behaviour of these quantifiers in various contexts.

Note too how sentences involving quantifiers and other logical operators are prone to scope ambiguities.

Quantifiers like 'everything/everyone' will range over different things from case to case. In ordinary usage, we typically need extraneous input to fix the current domain of quantification.

This gives us good reason to introduce formal QL languages with simply-behaved quantifiers, where there aren't scope ambiguities, and where domains of quantification are clearly fixed.

27 Quantifier-variable notation

The discussion in the last chapter suggests three desiderata for formal languages apt for exploring the logic of generality:

(1) We want to avoid the complicated multiplicity of ordinary-language ways of expressing general claims.
(2) We want to devise our notation so that the scope of a quantifying operation is always clear, leaving no possibility of scope ambiguities.
(3) We want to clearly fix what our quantifiers range over.

This chapter explains how we will design our QL languages to meet these requirements.

27.1 Quantifier prefixes and 'variables' as pronouns

(a) First, how are we going to avoid scope ambiguities? Well, as we saw in §26.3, we can usually keep things unambiguous even in ordinary language by rephrasing using *quantifier prefixes* like 'everyone is such that', 'some senior Republican senators are such that', 'there is some gift such that', 'every perceptual experience is such that', etc., where these are linked to later pronouns such as 'they' and 'it'. For example, we can disambiguate 'Every philosopher admires a certain book' by offering the alternative readings

(1) (Every philosopher is such that)(there is a book such that) they admire it,
(2) (There is a book such that)(every philosopher is such that) they admire it,

where these paraphrases are unambiguous – we just apply the rule that *a quantifier prefix governs what follows it*.

So we immediately have an attractive and quite natural model to emulate in designing languages for use in quantificational logic:

> In our formal QL languages, we will unambiguously render general propositions by using quantifier prefixes linked to later pronouns.

(b) How are we going to handle pronouns formally?

There are different uses of ordinary-language pronouns. There are, for example, demonstrative uses – as when I point across the room and say '*She* is a great logician' (I could have equivalently used a demonstrative and said '*That* woman is a great logician'). For my use of the pronoun here to be in good order, I must successfully pick

out a particular woman. Contrast the claim 'No woman is such that she enjoys being harassed'. In this case, 'she' plainly doesn't denote a particular woman! Nor does it in 'Every woman is such that she hates being harassed'. In these cases, the pronouns are not doing straightforwardly referential work but rather link back to, are 'bound' to, the previous quantifier prefix. Pronouns like this are classed as one kind of *anaphoric* pronoun (literally, they 'carry back'). But we will call them simply *bound* pronouns.

So let's sharpen our question: what shall our formal languages use as bound pronouns tied to quantifier prefixes?

(c) Suppose we want to informally render the more natural reading of

(3) Everyone loves a certain someone,

by using quantifier prefixes linked to later pronouns. Our gender-neutral singular pronoun is 'they'; but it won't do to write e.g.

(4) (Everyone is such that)(there is someone such that) they love them.

That is still ambiguous. Is it, so to speak, everyone or someone who is doing the loving?

We need, therefore, some way to indicate which pronoun is bound to which quantifier prefix. In examples (1) and (2) we could exploit the difference between the person-including 'they' and the impersonal 'it' to link them to the personal and impersonal quantifier prefixes respectively: but that's a special case. What can we do more generally?

One ordinary-language option is to use something like

(5) (Everyone is such that)(there is someone such that) the former loves the latter.

But that's not a very useful model to adopt – if only because it isn't easy keeping track of what counts as 'the former' or 'the latter' as we manipulate propositions while working through an argument. So here's a neat alternative trick we can use:

> We will henceforth adopt 'x', 'y', 'z' ... as additional pronouns.
>
> We then explicitly tag our quantifier prefixes with these new pronouns – as in '(Everyone x is such that)', '(Someone y is such that)' – in order to make it entirely clear which quantifier is bound to which later pronoun.

Hence instead of (5) we can write

(6) (Everyone x is such that)(there is someone y such that) x loves y.

Similarly, the other reading of (3) – as in 'Everyone loves a certain someone, namely Kylie' – gets unambiguously rendered as

(7) (There is someone y such that)(everyone x is such that) x loves y.

Compare and contrast e.g.

(8) (There is someone x such that)(everyone y is such that) x loves y.

which unambiguously says that someone (Jesus?) loves us all.

Of course, we have borrowed our new pronoun symbols 'x', 'y', 'z', etc., from the mathematicians, following *one* use they make of so-called *variables*. Consider, for example, the arithmetical truism

(9) For every number x, $x + 1 = 1 + x$

What does this say? The same as: 'every number is such that it plus one equals one plus it'. So we see that in this sort of case the mathematician's 'x' is indeed just doing the work of a pronoun like 'it'. And while using 'x's etc. as pronouns tied to more or less explicit quantifier prefixes may be most familiar from the maths classroom, there is nothing irredeemably mathematical about this usage.

(d) Quine famously wrote "Logic is an old subject, and since 1879 it has been a great one." Why that date? It is when the *Begriffsschrift* was published. And it is to Frege in this short book that we owe the insight that we can make the scope of quantifiers unambiguously clear by using a *quantifier-variable* notation where quantifier prefixes get linked to variables-as-pronouns. This will be our notation too.

27.2 Unary vs binary quantifiers

We have already said in §26.1 that we are going to build just two kinds of quantifiers into our QL languages, corresponding to 'every/all' and 'some/there is'. But we also noted that the ordinary-language versions of these come in two forms, unary and binary. And this still applies when we use quantifier prefixes linked to variables. So compare

(Everything x is such that) x is physical,
(Every philosopher x is such that) x is wise.

Or, generalizing, compare the schematic forms

(Everything x is such that) x is G,
(Every F, x, is such that) x is G.

The first quantifier construction takes one general term G, the second takes two general terms F, G to form a sentence. Again, we can call the first form of quantifier unary, the second binary. We might also say the second form involves a *restricted* quantifier, as the role of the F term here is to restrict what the initial quantifier prefix ranges over.

What, then, is the relationship between the unary and binary forms of quantifier here? A rather natural view might be that, in ordinary language (with or without added variables-as-pronouns), binary/restricted quantifiers are in fact the basic case. The apparently unary 'Everything is G' is really a special case of 'Every F is G', where F is replaced by the colourless, all-inclusive, 'thing'. Similarly, for example, 'Somebody is G' is a special case of 'Some F is G', where F is replaced by 'body' (meaning *person*).

However, it seems we can also go in the other direction, and instead treat the unary versions of the 'every' and 'some' quantifiers as basic. We then render the binary versions by equivalent propositions using just unary quantifier prefixes. Consider, for example,

(1) Every elephant has a trunk,

or the regimented version in quantifier-variable form

(2) (Every elephant x is such that) x has a trunk.

These look to be equivalent to the *quantified conditional*,

(3) (Everything x is such that)(if x is an elephant, then x has a trunk),

where now the quantifier is unary. Likewise, consider the generalization

(4) Some elephant trampled the grass,

or the regimented version

(5) (Some elephant x is such that) x trampled the grass.

These look to be equivalent to the *quantified conjunction*

(6) (Something x is such that)(x is an elephant and x trampled the grass).

We will need to say more about these claimed equivalences, and about why we get a conditional in (3) and a conjunction in (6) (see §29.1). But for the moment, let's assume that we can – without mangling content too much – exchange binary 'every' and 'some' for unary quantifiers plus connectives.

So which way shall we jump? Do we develop our formal quantifier-variable treatment of 'every/some' general propositions in a way that treats the binary versions as basic? Or do we treat the unary quantifiers as basic?

Taking the second line means being less faithful to the surface forms of natural language. However, it *does* have the virtue of keeping our formal quantifiers particularly simple, and so this is the line adopted by most logicians ever since Frege. So:

> We will only build *unary* quantifiers into our QL languages – just formal versions of '(everything x is such that)' and '(something y is such that)', etc. We then express restricted quantifications by using conditionals and conjunctions inside the scope of these unary quantifiers.

Do note, however, that privileging unary quantifiers like this *is* an independent extra decision, over and above the fundamental decision to adopt a quantifier-variable notation.

27.3 Domains

(a) We have decided then that our formal QL languages will only have built-in 'every' and 'some' quantifiers in their unary forms. What do these quantifiers range over?

As we noted in the last chapter, ordinary discourse often leaves it unspoken exactly who or what we are quantifying over. It is left up to context and interpretative charity to settle what counts as 'everything' or 'everyone'. And the universe of discourse is often allowed to shift as conversation progresses. By contrast, we will want everything in our formal languages to be explicit and stable, with nothing left to guesswork.

The standard approach is to take the quantifiers in a QL language to all run over the *same* unshifting domain. And a further standard stipulation is that domains always contain at least one object. Hence

> To interpret a QL language, we fix at the outset, once and for all, one common non-empty domain for its quantifiers. We do this by giving a description D of the domain in the glossary for the language (where at least one thing satisfies D).

(b) The one-common-domain convention buys us clarity and simplicity – though at the cost of some artificiality, including some departure from mathematical practice.

Mathematicians using semi-formal language often use different quantifiers with different domains, associated with different sorts of variables. For example, an arithmetician might typically use lower case variables for numbers, and upper case variables for sets of numbers; an algebraist might typically use letters from early in the alphabet for scalars, and letters from the end of the alphabet for vectors. Simultaneously using quantifiers tied to distinct sorts of variables to range over distinct domains is very natural. However – and to repeat, this is the usual convention – our official QL languages will have just *one* sort of variable ranging over *one* inclusive domain. Hence, when we want to quantify over different sorts of things in a single QL language, we will again have to explicitly restrict our all-inclusive quantifiers using connectives.

(c) Domains can be small – comprising just, say, *the students signed up to the logic class*. However, as we said, we will ban completely empty domains. In another words, we assume that a QL language isn't talking about nothing at all. We return to say more about this rather natural stipulation in §**??**.

Domains can also be very large – as when our quantifiers range over *everyone now living*, or over all *elementary particles* or all *natural numbers* or all *sets*. Indeed, we can allow a language's universe of discourse to be absolutely everything – every object there is, of any sort – *if* that idea makes sense. However, that's a very big 'if'! (To get the flavour of *one* reason why there might be a problem here, consider this thought, plausible if we take sets seriously enough. Given any universe of objects, however many there are, there is always *another* object, namely the set of all the objects collected together so far. So any proposed totality of objects is always extensible by another object, and there is no fixed totality of *all* objects. Or so one story goes. But we can't tangle with this troublesome line of argument here.)

27.4 Quantifier symbols

(a) We now take one more step, moving from stilted-English-using-variables-as-pronouns towards something even closer to a standard logical QL language. We introduce the *quantifier symbols '∀' and '∃'*, as follows:

> Instead of writing '(everything/everyone x is such that)', we will simply use the very terse notation '$(\forall x)$', with the rotated 'A' reminding us that this can also informally be read as 'for all x'.
>
> And instead of '(something/someone y is such that)' or '(there is something/someone y such that)' we will use '$(\exists y)$'. Here, the rotated 'E' reminds us that this can also informally be read as 'there exists y such that'.

Assume that we are working in a context where the quantifiers range over, say, all people. Then our three examples from §27.1, there numbered

(6) (Everyone x is such that)(there is someone y such that) x loves y
(7) (There is someone y such that)(everyone x is such that) x loves y
(8) (There is someone x such that)(everyone y is such that) x loves y,

can now be very neatly abbreviated in turn as follows:

(6′) $(\forall x)(\exists y)$ x loves y
(7′) $(\exists y)(\forall x)$ x loves y
(8′) $(\exists x)(\forall y)$ x loves y.

(b) Keeping the same universe of discourse, now consider how we can express restricted generalizations using our new notation. Take e.g.

(1) Everyone in the class has arrived.

Then, following the suggestion made in §27.2, we can render this as

(2) (Everyone x is such that)(if x is in the class, then x has arrived),

which now becomes

(3) $(\forall x)$(if x is in the class, then x has arrived),

where '$(\forall x)$' still ranges over all people. Taking another step of symbolization, this is arguably equivalent to

(4) $(\forall x)(x$ is in the class $\rightarrow$ x has arrived).

The next sentence, however, is potentially ambiguous

(5) Everyone in the class has not arrived.

We can, in some contexts, understand this with 'not' having wide scope, i.e. we can understand (5) as conveying the message unambiguously expressed by

(6) $\neg(\forall x)(x$ is in the class $\rightarrow$ x has arrived).

But in other contexts, we will understand (5) as meaning

(7) $(\forall x)(x$ is in the class $\rightarrow \neg$ x has arrived).

In (6) and (7) the relative scopes of the negation and quantifier are now transparently clear.

Likewise,

(8) Some footballer deserves great riches,

which we can render using a prefixed unary quantifier as

(9) (Someone x is such that)(x is a footballer and x deserves great riches),

can now be partially symbolized as

(10) $(\exists x)(x$ is a footballer and x deserves great riches),

or equivalently, taking a further step, as

(11) $(\exists x)(x$ is a footballer $\wedge$ x deserves great riches).

And what about the following sentence, which taken out of context is ambiguous?

(12) Some footballer does not deserve great riches.

We can half-symbolize the two possible readings along the following lines:

(13) $(\exists x)(x$ is a footballer $\wedge \neg$ x deserves great riches),
(14) $\neg(\exists x)(x$ is a footballer $\wedge$ x deserves great riches).

Again, in (13) and (14) the scopes of the negation and quantifier are now unambiguous.

(c) The sort of unholy mixture of English and logical symbolism in e.g. (8′), (7) and (14) is often called *Loglish*. It easily understood, given a grasp of the connectives, and a grasp of the new quantifier symbols in their abbreviatory role. And as we will see in the coming chapters, going via varieties of Loglish as a half-way house greatly eases the transition between ordinary language and fully formalized QL languages.

27.5 Unnamed objects

Now a simple point, but one that we should stress: not everything in a domain need have a fixed proper name! This has two important implications.

(a) Let's use $(\forall v)A(v)$ informally to represent a Loglish sentence starting with the quantifier prefix $(\forall v)$, where $A(v)$ is an expression with one or more occurrences of the variable v. Let $A(n)$ be result of replacing all the occurrences of v in $A(v)$ with the proper name n. Then we will say that $A(n)$ is an *instance* of the quantified sentence $(\forall v)A(v)$. For example '(Ludwig is in the class → Ludwig has arrived)' might be an instance of '$(\forall x)$(x is in the class → x has arrived)' .

Now, what holds of everything in a domain holds for any particular named thing (for we will assume names currently in use do indeed refer to things in the current domain). So from $(\forall v)A(v)$ we can infer any corresponding particular instance $A(n)$.

But the reverse is false. It could be that for any available name n in the language, $A(n)$ is true but $(\forall v)A(v)$ is still false because some unnamed object fails to satisfy the condition expressed by A. For example, it may be that – as the Psalmist sings – the Lord "telleth the number of the stars; he calleth them all by their names". But *we* certainly don't have names for all the stars. And if we say 'All stars contain iron' (for example) – i.e. '$(\forall x)$ x contains iron', where the quantifier ranges over stars – then our claim wouldn't be made true just by the fact that the *named* stars all happen to contain iron.

(b) Similarly, let $(\exists v)A(v)$ informally represent a Loglish sentence starting with the quantifier prefix $(\exists v)$. And we will say that $A(n)$ is an instance of the quantified sentence $(\exists v)A(v)$ too.

Now, since what holds of a particular named thing holds of something in the domain (keeping the assumption that names currently in use do indeed refer to things in the current domain), from any sentence $A(n)$, where n is a name for some object in the domain, we can infer $(\exists v)A(v)$.

Again the reverse is false. It could be that $(\exists v)A(v)$ is true, even if each particular instance $A(n)$ is false (where n is a name already in our language). Consider, for example, the true claim '$(\exists x)$ x is a nameless star'!

27.6 A variant notation

(a) Consider again the quantified sentence

(1) $(\forall x)(\exists y)$ x loves y

which corresponds to one reading of 'Everyone loves a certain someone'. It is important to emphasize that the particular choice of variables in use here is quite arbitrary. The

role of the variables is simply to tie the quantifier prefixes to the places either side of 'loves'. To express the same message we could therefore equally well use e.g.

(2) $(\forall y)(\exists z)$ y loves z, *or* $(\forall y)(\exists x)$ y loves x, *or* $(\forall z)(\exists y)$ z loves y, *or* ...

Here is a variant notation which ties quantifier prefixes to places graphically. Take the predicate '① loves ②'. Then instead of (1) or any of (2) we could write simply

(3) ∀ ∃ ① loves ②

And similarly, instead of using e.g.

(4) $(\exists y)(\forall x)$ x loves y

to express the other reading of 'Everyone loves a certain someone', we could write

(5) ∃ ∀ ① loves ②

For another example, consider again

(6) $(\forall x)$(if x is in the class, then x has arrived).

We can think of this as involving the complex predicate 'if ① is in the class, then ① has arrived' (using a repeated counter to indicate that the gaps are to be filled in the same way – see §25.2(d)). And we could alternatively express the message that everyone satisfies this predicate by writing e.g.

(7) ∀ (if ① is in the class then ① has arrived)

Of course, this braces-and-gaps graphical notation is difficult to typeset, and it gets difficult to read when more than one prefixed quantifier symbol gets tied to multiple later gaps. No wonder, then, that we will stick to the now conventional quantifier/variable notation. Still, note that the use of a variable to link a quantifier to some place(s) in an expression is really just a variant on our graphical notation. For example,

(1) $(\forall x)(\exists y)$ x loves y,

(3) ∀ ∃ ① loves ②

are to be explained as ultimately meaning just the same. This nicely re-emphasizes that the particular choice of variable-letters is irrelevant, so long as we keep fixed the pattern of ties from quantifier prefixes to places-occupied-by-variables.

(b) The graphical notation also vividly brings out another important point. In the graphical notation, the quantifier symbol '∀' by itself doesn't yet implement a generalization – it needs to be tied to slots in the following gappy expression. It is '∀'-plus-the-ties which does the work.

It is exactly similar when we use variables. '∀' without its variable does not yet implement a generalization. Nor does the quantifier prefix '$(\forall x)$' by itself. It is the prefixed quantifier plus the later linked variable(s) – the combination that together forms we can call a quantifier operator – which does the work.

Now, linguists use 'quantifier' to mean expressions in ordinary language like 'every' and 'some' etc. The nearest equivalent in Loglish – i.e. the simple expressions which determine the *kind* of generalization we are making – are the quantifier symbols '∀' and

'∃': and some writers do indeed call these simply 'quantifiers'. If we alternatively hijack the word 'quantifier' for the expression that does the work of forming a generalized claim, then – as we've just noted – it is the quantifier prefix *plus* its linked variables which really deserves the label. However, it is also quite usual to call the quantifier prefix '$(\forall x)$' by itself a 'quantifier', even though that arguably labels the wrong thing.

But let's not get hung up on the terminological point. Let's relax and agree to cheerfully use the word 'quantifier' in whichever way is convenient in context.

27.7 Summary

At the beginning of the chapter, we listed three desiderata to guide the design of our formal QL languages. We can now indicate how we are going to respond to each one.

A primary goal is to devise our notation so that the scope of a quantifying operation is always clear and unambiguous. We will meet this central requirement by using quantifier prefixes tied to variables-as-pronouns in our QL languages.

It is also important to avoid the complicated multiplicity of ordinary-language ways of expressing general claims. So we are going to restrict ourselves to just two types of quantifier-prefixes in QL languages, corresponding to the ordinary-language 'every' and 'some' in their unary versions. We will symbolize these using '∀' (tagged with a variable) for 'every', and '∃' for 'some'.

We want to clearly fix what our quantifiers range over. We will do this by the crude-but-effective method of simply stipulating, when setting up a formal QL language, the common domain of quantification for its quantifiers.

So our QL languages have only unary quantifiers, and these all run over the same domain. It remains to be seen just how expressive the resulting formal languages are. How well, for example, can we capture the content of ordinary language binary or restricted quantifications?

28 QL languages

The previous chapter explained the motivation for adopting some style of quantifier-variable notation for expressing generality. In this chapter, we explore QL languages incorporating this sort of notation.

28.1 QL languages introduced

The informal Loglish of §27.4 introduces a symbolic quantifier-variable notation to express generality and recycles the familiar symbols for the connectives; but it keeps English for non-logical vocabulary. Our formal QL languages will also use the same connectives and the quantifier-variable notation, but will now use symbols for names and predicates too.

So we will symbolize

(1) Romeo loves Juliet

by the likes of

(2) Lrj

using lower case letters like 'r' and 'j' for names, upper case letters like 'L' for predicates, and following a standard convention where we write predicates *before* the names or variables they apply to. Then, instead of

(3) (Everyone x is such that)(there is someone y such that) x loves y

or the Loglish version

(4) $(\forall x)(\exists y)$ x loves y,

in a QL language we will write e.g.

(5) ∀x∃y Lxy.

And instead of

(6) (Everyone x is such that)(if x is in the class then x has not arrived)

or the Loglish version

(7) $(\forall x)(x$ is in the class $\rightarrow \neg\, x$ has arrived),

a QL language will provide something like

(8) ∀x(Cx → ¬Ax).

Let's take one more example. The claim

(9) Every philosopher admires some logician

is potentially ambiguous. The less natural reading (though the one that would be mandated if the claim was followed by 'namely Aristotle') can be unambiguously rendered like this, using unary quantifier prefixes:

(10) (There is someone x such that)(x is a logician and (everyone y is such that)(if y is a philosopher then y admires x)).

Using Loglish shorthand, this becomes

(11) $(\exists x)(x \text{ is a logician} \wedge (\forall y)(y \text{ is a philosopher} \rightarrow y \text{ admires } x))$.

And this goes into a suitable QL language – with the predicate letters now appropriately (re)interpreted – as

(12) $\exists x(Lx \wedge \forall y(Py \rightarrow Ayx))$.

As these examples show, then, once you have grasped the quantifier-variable notation in its informal Loglish guise, it is just a small step to understand formal QL languages.

The rest of this chapter says more about the syntax and interpretation of such languages. Along the way, we will need to silently make choices about how to handle some matters of detail. We will return to briefly note some choice-points in §32.1 and indicate why, in the end, it doesn't matter too much how we jump. (We can leave a comparison with some other ways of handling details for an Appendix.)

28.2 Names, predicates and atomic wffs in QL: syntax

(a) The atomic wffs of a QL language are formed from *predicates* with fixed arities combined with the right number of *terms* – see §25.1(b) and §25.2(b).

It will be important that our QL languages all come with an indefinitely large supply of terms in the guise of so-called dummy names. But we don't need them yet, and we will also leave aside functional terms. For the moment, the only terms we are going to be introducing are *proper names*.

So, here at the outset, the non-logical vocabulary of a QL language – the vocabulary which can vary from language to language – will comprise just names and predicates.

We want to keep our symbolism uncluttered. So we will typically use *single letters* for these non-logical building blocks of QL languages, just as we used single letters for the non-logical building blocks of PL languages. And, as introduced in the last section, there is an entirely standard convention of using *lower case* letters from mid-alphabet for proper names and using *upper case* letters for predicates. This convention is prefigured in our earlier informal use of letters like 'n' to stand in schematically for names and the likes of '(is) F' to stand in for predicates.

Hence, in summary:

> The built-in non-logical vocabulary of a QL language can include
>
> *proper names*, usually letters like: m, n, o, …;
> *predicates*, usually letters like: F, G, …, L, M, …, R, S, ….

Proper names are *terms*.

Each predicate is assigned a fixed arity and will take a fixed number of terms to form an atomic wff.

(b) Next, how do we actually put together a predicate and the right number of terms to form an atomic wff in a QL language? In English we can put predicates *after* names (as in 'Felix is scary') or *between* names (as in 'Romeo loves Juliet' or 'Jo prefers Jack to Jill'). In the informal schematic notation introduced in Chapter 1, we represented the simplest name/predicate sentences by the likes of 'n is F', again following the order of ordinary language, and inserting an 'is' for readability. In QL languages, however, we conventionally put the predicate *before* the names (or other terms), and we don't need the equivalent of 'is' to glue them together. Hence, our basic rule will be simply this:

In a QL language, a predicate of arity k followed by k terms from the language forms an atomic wff – perhaps, for example: Fm, Lnm, Loo, Rnmo, ...

(c) Note, however, this way of structuring atomic wffs is no more than a default convention, blessed by tradition but having no intrinsic significance. Later we will allow some exceptions to the predicates-first rule. We will also allow the use of non-letter symbols for names and/or predicates. (That will mean, for example, that we can still write the likes of '$0 < 1$' as an atomic sentence in a formal QL language for arithmetic.)

28.3 Names, predicates and atomic wffs in QL: interpretation

Now for interpreting the QL symbolism we have so far. We again follow the lead of our discussion of ordinary-languages names and predicates in Chapter 25.

(a) QL names refer to objects (in that broad sense of 'object' that can include such disparate things as people and mountains, stars and molecules, numbers and sets, etc.). QL predicates express conditions for objects, or tuples of objects, to satisfy.

And how do we fix the reference of a name or fix the condition expressed by a predicate? By giving a glossary, which for us will be in English. So:

A glossary for a QL language interprets a QL proper name by associating it with an (in context, unambiguous) English name or other expression referring to some unique object.

A glossary interprets a QL predicate of arity k by associating it with some English expression giving the condition which a k-tuple of objects needs to satisfy if the formal predicate is to be true of those objects.

An atomic QL wff formed by a k-ary predicate followed by k proper names says that the named objects taken in order satisfy the condition expressed by the predicate.

(b) One important proviso. The reference of a name in some language must be an object in that language's domain of quantification. This is to ensure that, given as premiss a wff saying that everything (in the domain) is F, we can infer that any particular named

object is F – and likewise, given that a named object is F, we can infer the quantified wff saying that something (in the domain) is F.

(c) Another important proviso. The condition expressed by a predicate must be sharp enough for it to be quite determinate whether or not an object satisfies the condition. This ensures that the sentences of QL languages will be like the wffs of PL languages in having definite truth values (compare §10.3).

Ordinary language has vague predicates with borderline cases: it is very natural to say, for example, that it might be indeterminate whether Bill is bald, neither true nor false. By our determinacy assumption, QL languages must lack predicates like this. So, when we interpret a formal predicate by associating it with an English predicate, we have to quietly assume that any vagueness in the English has been tidied away (perhaps by drawing artificially sharp boundaries) or can just be ignored.

(d) Finally, a neat little addition! *We will allow the case where a formal predicate has arity zero.* By our syntactic rule, a predicate of arity zero followed by zero terms forms an atomic wff. So a predicate of arity zero just *is* an atomic wff, comparable to an atom of PL. Hence we will interpret a predicate of arity zero by assigning it a sentence in a glossary, again just like a PL atom.

It obviously makes no difference at all whether we say that a QL language (i) can have predicates of various arities from one up, plus propositional letters as well, or (ii) can have predicates of various arities from zero up. The second way of putting it is just that little bit neater!

28.4 One example: introducing QL$_1$

(a) At the level of syntax, we can specify the non-logical vocabulary of a simple QL language (without function expressions) by just listing its built-in proper names and listing its built-in basic predicates, fixing the arity of each predicate as we go. Then, at the level of semantics, we give a glossary for these names and predicates.

We can neatly package the syntactic and semantic stories together, like this:

In the language QL$_1$, the *proper names* are just

m: Socrates,
n: Plato,
o: Aristotle.

and the *predicates* are just

F: ① is a philosopher,
G: ① is a logician,
H: ① is wise,
L: ① loves ②,
M: ① is a pupil of ②,
R: ① prefers ② to ③.

We take the explicit arity of the interpreting informal predicate to fix the arity of the corresponding formal predicate. And we then match the j-th place after the formal predicate to the informal predicate's place marked with the j-th counter in the obvious way. For example, 'R' followed by three names says that the first named individual prefers the second to the third.

(b) So here are seven atomic wffs from QL_1 (formed according to our construction rule) together with their interpretations (following our interpretation rule, given the glossary):

Fm:	Socrates is a philosopher.
Hn:	Plato is wise.
Lnm:	Plato loves Socrates.
Loo:	Aristotle loves himself.
Mon:	Aristotle is a pupil of Plato.
Rnmo:	Plato prefers Socrates to Aristotle.
Romo:	Aristotle prefers Socrates to himself.

28.5 Adding the connectives

(a) Next, we take over from PL languages the now familiar connectives plus the absurdity sign. Both the syntactic and the interpretative rules for these symbols are just as you would expect:

> If α and β are wffs of a given QL language, so too are $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \rightarrow \beta)$. The absurdity sign '$\bot$' is also a wff of any QL language.
>
> The connectives are to be interpreted just as before – as (tidied-up) negation, conjunction, inclusive disjunction, and the material conditional.
>
> The absurdity sign can be thought of as expressing a non-specific contradiction.

Given these newly added rules for wff-building with connectives, the following are therefore also wffs of QL_1, with the associated interpretations:

$\neg$Fm:	Socrates is not a philosopher.
(Go $\rightarrow$ Ho):	If Aristotle is a logician, he is wise.
(Hn $\wedge$ Lmn):	Plato is wise and Socrates loves him.
(Lom $\wedge \neg$Lno):	Aristotle loves Socrates but Plato doesn't love Aristotle.
(Rnmo $\rightarrow$ (Lom $\wedge \neg$Lon)):	If Plato prefers Socrates to Aristotle, then Aristotle loves Socrates but not Plato.

(b) We should strongly emphasize that connectives in a QL language do remain essentially propositional or sentential connectives – in the sense that, when we construct a wff stage-by-stage, a connective always gets introduced as connecting *wffs*.

This means, in particular, that we must render e.g. 'Plato and Aristotle are philosophers' into QL_1 as

(Fn $\wedge$ Fo)

and not, repeat *not*, as the ill-formed

$F(n \wedge o)$.

In English, as we noted in §8.2, 'and' can connect two names as well as two sentences. In QL languages it is quite different: '$\wedge$' cannot connect names.

Similarly, to render 'Aristotle is a logician and is wise' into QL_1, we must use

$(Go \wedge Ho)$

and not, repeat *not*, the ill-formed

$(G \wedge H)o$.

Again, unlike the English 'and', the QL connective '$\wedge$' cannot connect predicates.

28.6 Syntax for the quantifiers

So far, so straightforward. Now for the key step, adding quantifiers to our formal languages. Syntax first.

(a) We add to the logical vocabulary of QL languages as follows:

> Every QL language has an unlimited supply of *variables*, starting
>
> $x, y, z, \ldots$.
>
> It also contains the two *quantifier symbols* (read, roughly, 'for all', 'for some')
>
> $\forall, \exists$.
>
> We tag a quantifier symbol with a variable to get a quantifier prefix, or *quantifier* for short, as in
>
> $\forall x, \forall y, \forall z, \ldots$
>
> $\exists x, \exists y, \exists z, \ldots$
>
> Quantifiers of the first kind are called *universal* quantifiers; quantifiers of the second kind are called *existential* quantifiers.

Note that, unlike in our informal Loglish, we will *not* put brackets round quantifier prefixes in our official QL languages (we don't really need them, and the more austere bracket-free notation is nowadays at least as common, so let's learn to love it).

When we come to specifying QL languages more carefully, we will of course need to be more rigorous about what counts as a variable, and not just give an open-ended list. But we need not worry about that yet. The key point is that symbols for variables are to be sharply distinguished from symbols for proper names (and other terms).

(b) So how do we form quantified QL wffs? The key idea is that variables can take the place of names or other terms, but they serve as bound pronouns, so a variable must always be introduced along with a quantifier prefix tied to it. So, take a formal wff containing one or more occurrences of a some proper name or other singular term. Replace all the occurrences of that term with a variable not already in the wff. Prefix a formal quantifier using the same variable. And we get a new QL wff.

It will help to restate the rule symbolically. So here are a couple of conventions for the use of schematic letters (natural conventions, at least given our policy of using Greek letters in schemas representing formal expressions!).

> 'τ' – *tau*, the Greek *t* – will be used to stand in for a term.
>
> 'ξ', 'ζ' – *xi*, *zeta*, the Greek *x*, *z* – will be used to stand in for formal variables.

Then here is our syntactic rule for forming quantified wffs.

> Suppose $\alpha(\tau)$ stands in for a QL wff involving one or more occurrences of the term τ, and $\alpha(\xi)$ is the expression which results from replacing the term τ throughout $\alpha(\tau)$ by a variable ξ new to that wff.
>
> Then $\alpha(\xi)$ is *not* a wff. However, $\forall\xi\alpha(\xi)$ and $\exists\xi\alpha(\xi)$ *are* QL wffs.

Schematic notation like '$\alpha(\tau)$' is common. But note, it *doesn't* signify that the represented expression contains brackets, only that it contains the term τ!

Some handy shorthand: we will say that a wff $\forall\xi\alpha(\xi)$ formed from $\alpha(\tau)$ in this way is formed by *universally quantifying on* τ. Similarly, $\exists\xi\alpha(\xi)$ is formed from $\alpha(\tau)$ by *existentially quantifying on* τ. (And remember, we said that every QL language will have an unlimited supply of terms in the form of dummy names; so we aren't going to run out of terms to quantify on in order to construct arbitrarily complicated quantified wffs. But let's not fuss about that now.)

(c) To make all this really clear, we will step slowly through some very easy examples. Start with the QL_1 wff

(1) $(\mathsf{Fm} \rightarrow \mathsf{Hm})$.

Now we quantify on the only term. We replace the term (in this case, name) 'm' by the variable 'x' and prefix the result with '∀x'. We get the wff

(2) $\forall\mathsf{x}(\mathsf{Fx} \rightarrow \mathsf{Hx})$.

Equally, starting from

(3) $(\mathsf{Fm} \wedge \mathsf{Lmn})$

and applying our construction rule, we can existentially quantify on 'm' to get e.g.

(4) $\exists\mathsf{z}(\mathsf{Fz} \wedge \mathsf{Lzn})$.

We can proceed another step from (4), and now replace the name 'n' with a variable. We have to choose a variable that doesn't occur in the wff we are building from, so we can't use 'z' again. But we could pick, e.g., 'x'. Then we prefix a quantifier symbol tagged with *this* new variable; say '∃x'. Which gives us the wff

(5) $\exists\mathsf{x}\exists\mathsf{z}(\mathsf{Fz} \wedge \mathsf{Lzx})$.

Similarly, starting from the QL_1 wff

(6) Lmn

we can quantify on 'n' and then 'm' to form

(7) $\exists y\, Lmy$,

(8) $\forall x \exists y\, Lxy$.

Or – starting from (6) again – we can replace names in a different order and/or use different variables and/or use different quantifier symbols. We could form, for example,

(9) $\forall x\, Lxn$,

(10) $\exists y \forall x\, Lxy$.

Equally, we could get from (6) to

(11) $\exists y \forall x\, Lyx$, $\forall y \forall x\, Lyx$, $\exists z \exists y\, Lzy$, ...

Note that our construction rule doesn't allow us to get from (6) to e.g.

(12) $\exists y\, Lyy$, $\forall z\, Lzz$, ...

(Think why not!) However, those expressions *are* wffs, as they can be formed by quantifying on 'm' in the wff

(13) Lmm.

Since 'Hm' is a wff as well as (7), we can use a connective to construct another wff

(14) $(Hm \rightarrow \exists y\, Lmy)$.

Then, universally quantifying on 'm', we get e.g.

(15) $\forall x(Hx \rightarrow \exists y\, Lxy)$.

And since '$\forall x\, Mxn$' is of course a wff alongside (9), so too is their disjunction

(16) $(\forall x\, Lxn \vee \forall x\, Mxn)$;

and existentially quantifying on 'n' we get, e.g.

(17) $\exists y(\forall x\, Lxy \vee \forall x\, Mxy)$.

And so on it goes – with many more examples of wffs to come in the next chapter!

(d) Now for some non-wffs. Expressions like the following

(18) Fz, Lzn, Myx, $Rxoy$, $(Fm \rightarrow Gz)$, $\exists x\, Lxy$

are *not* QL_1 wffs. Expressions with variables dangling free, not bound to preceding quantifiers, do *not* count as wffs according to our construction rules.

And, given our ban on reusing variables already in a given wff when applying a quantifier, we *can't* get from (7) to the expression

(19) $\exists y \exists y\, Lyy$.

Note too that there is no way to prefix another quantifier to the likes of (8). We can't form an expression with a dangling quantifier, not tied to any later variable, as in

(20) $\forall z \forall x \exists y\, Lxy$.

Neither of the expressions (19) and (20) will count as wffs.

And because the wff (14) already contains the variable 'y', we can't quantify it using the same variable again. This is another non-wff:

(21) $\forall y(Hy \rightarrow \exists y\, Lyy)$.

(e) To repeat, 'Fz' and 'Lzn' are *not* wffs. So inside the wff (4) '$\exists\mathsf{z}(\mathsf{Fz} \wedge \mathsf{Lzn})$' the connective '$\wedge$' does *not* connect wffs. How does this square with our stern words about connectives in §28.5(b)?

Well, note that '$\exists\mathsf{z}(\mathsf{Fz} \wedge \mathsf{Lzn})$' is constructed by quantifying on a name in a wff like (3) '$(\mathsf{Fm} \wedge \mathsf{Lmn})$'. And in (3) the connective *does* connect wffs.

The point generalizes. As we put it before, in building up a wff, a connective will indeed first get introduced as connecting whole wffs. That is why we can say that connectives in a QL language do remain essentially sentential connectives.

(f) We will refine our account of QL syntax later. But perhaps we should add a quick remark now.

We just noted that e.g. the QL_1 wff '$\exists\mathsf{z}(\mathsf{Fz} \wedge \mathsf{Lzn})$' can be constructed by quantifying on 'm' in the wff '$(\mathsf{Fm} \wedge \mathsf{Lmn})$'. But equally it can be constructed by quantifying on 'o' in the wff '$(\mathsf{Fo} \wedge \mathsf{Lon})$'. Hence – at least on the story so far – a quantified will *not* have a unique possible constructional history/parse tree. Does this matter?

No. Different constructional histories for a quantified wff will structurally look just the same: they will just differ in the names (or other terms) that occur earlier in the history of a wff and which then get quantified away. Therefore our story still doesn't allow structurally ambiguous wffs. And so, for example, it remains determinate what the main logical operator (connective or quantifier) of a quantified wff is – i.e. there is no ambiguity about what is the *last* operator introduced in a constructional history. For example, there is no ambiguity about whether a wff has the form $\forall\xi\alpha(\xi)$ or has the form $\exists\xi\alpha(\xi)$. So we needn't worry about the uniqueness issue now (but see §**??**).

28.7 Interpreting the quantifiers

(a) Now for matters of interpretation. As we said, we need to begin like this:

> To interpret a QL language we must specify its (non-empty) domain. We do this in the glossary for the language by giving some *domain description*.

The assigned domain for a QL language can be any set of objects, large or small with that single proviso that there must be at least one object in the domain. We can then continue:

> Suppose that, in a given QL language, the term τ refers to the object o and suppose, given the interpretation of other vocabulary, the sentence $\alpha(\tau)$ involving that term says that this object o satisfies a certain condition, says that o is C.
>
> Universally quantifying on τ, $\forall\xi\alpha(\xi)$ says that everything is C – meaning everything in the relevant language's universe of discourse, of course.
>
> Existentially quantifying on τ, $\exists\xi\alpha(\xi)$ says that something, at least one thing, is C – meaning something in the relevant universe of discourse again.

This account should fit nearly with your prior understanding of the quantifier prefixes in their Loglish guise.

(b) We will illustrate this interpretative scheme by working through some examples from our mini-language QL_1. But first we need to complete the interpretation key for this language by fixing what its quantifiers range over. So we give a domain description in the language's glossary:

> The domain of quantification for QL_1: people, past and present.

Let's start off, then, with the wff

(1) $(\mathsf{Fm} \to \mathsf{Hm})$.

Given our glossary for the language, this says that if Socrates is a philosopher, then Socrates is wise. In other words, Socrates is such that if he is a philosopher, then he is wise – or for short, Socrates is *C*. Then

(2) $\forall \mathsf{x}(\mathsf{Fx} \to \mathsf{Hx})$

says that *everyone* (i.e. everyone past and present) is *C*: i.e. everyone is such that if they are a philosopher then they are wise. Equivalently, all philosophers are wise.

Similarly,

(3) $(\mathsf{Fm} \wedge \mathsf{Lmn})$

says that Socrates is a philosopher and Socrates loves Plato. In other words, it says Socrates is such that he is a philosopher and he loves Plato – for short, recycling the informal notation, Socrates is *C*. And then

(4) $\exists \mathsf{z}(\mathsf{Fz} \wedge \mathsf{Lzn})$

says that there is *someone* who is *C*: i.e. there is someone who is a philosopher and loves Plato. Equivalently, some philosopher loves Plato.

Next, take the simple wff

(5) Lmn.

This of course says that Socrates loves Plato. Hence, when we existentially quantify into the place occupied by 'n' as in

(6) $\exists \mathsf{y}\, \mathsf{Lmy}$,

this says that there is someone who has the property which (5) ascribes to Plato. In other words, there is someone loved by Socrates – or equivalently, Socrates is such that they love someone.

By universally quantifying into the place occupied by 'm', we can then say the same about everyone. Therefore

(7) $\forall \mathsf{x} \exists \mathsf{y}\, \mathsf{Lxy}$

means that everyone is such they love someone.

Needless to say, however, we don't really need to build up to the interpretation of (7) laboriously like this! In practice, since the QL and the Loglish quantifiers effectively mean the same, we can just directly transcribe (7) into Loglish as

(8) (Everyone *x* is such that)(there is someone *y* such that) *x* loves *y*.

And, once we've fixed the domain, this stilted not-quite-English is already perfectly understandable as it is.

(c) Let's work through another series of examples.

(9) (Fm ∧ Romn)

says that Socrates is a philosopher and Aristotle prefers him to Plato. So what about

(10) ∃y(Fy ∧ Royn)?

This says that someone has the property that (9) attributes to Socrates, i.e. there is someone who is a philosopher whom Aristotle prefers to Plato. Hence

(11) (Go → ∃y(Fy ∧ Royn))

says that if Aristotle is a logician, there's some philosopher he prefers to Plato. Now suppose we want to say about everyone what this says about Aristotle. Then consider

(12) ∀x(Gx → ∃y(Fy ∧ Rxyn)).

This says: everyone is such that, if they are a logician, then there's some philosopher they prefer to Plato. In more natural English, every logician prefers some philosopher to Plato.

But of course, again we won't in practice go through all this step-by-step interpretative palaver when faced with a wff such as (12). Again, the thing to do is just transcribe it into Loglish as a half-way house, as in

(13) (∀*x*)(if *x* is a logician, then (∃*y*)(*y* is a philosopher and *x* prefers *y* to Plato)).

which in turn unpacks as

(14) (Everyone *x* is such that)(if *x* is a logician, then (there is someone *y* such that)(*y* is a philosopher and *x* prefers *y* to Plato)).

which is very stilted not-quite-English but entirely understandable all the same!

And those are enough elementary examples for the moment. There will be *lots* more examples of translating in and out of a QL language via Loglish in the following chapters.

28.8 Quantifier equivalences

(a) Given our glossary for QL_1, and the rule for interpreting quantified wffs,

(1) ∀x¬Fx

says that everyone is a non-philosopher – i.e. no one is a philosopher. Therefore

(2) ¬∀x¬Fx

says that it isn't the case that no one is a philosopher – in other words, there is someone who *is* a philosopher. Hence this says the same as

(3) ∃xFx.

Similarly, since

(4) ∃x¬Fx

in QL_1 means that someone is not a philosopher, its negation

(5) ¬∃x¬Fx

says that it isn't the case that there is someone who is not a philosopher – in other words, everyone *is* a philosopher. Hence (5) is equivalent to

(6) ∀xFx.

And the two equivalences here obviously generalize. Use 'ξ' as before to indicate a variable, and now for brevity use 'φ' (*phi*) for some suitable wff-completing expression containing that variable, then

> In QL languages, wffs of the forms $\neg\forall\xi\neg\varphi$ and $\exists\xi\varphi$ (with the same completing expression φ) are equivalent, i.e. are true in just the same circumstances.
>
> Likewise, wffs of the forms $\neg\exists\xi\neg\varphi$ and $\forall\xi\varphi$ are equivalent.

(b) In short, universal and existential quantifiers are interdefinable using negation, just as conjunction and disjunction are interdefinable using negation (see §13.1).

The parallel should not be in the least surprising. If we pretend for a moment that everything has a name, then we can think of '∃xFx' (for example) as in effect a big disjunction like 'Fm ∨ Fn ∨ Fo ∨ . . .'. By a version of one of De Morgan's Laws, this big disjunction is equivalent to the *negation* of the big conjunction '¬Fm ∧ ¬Fn ∧ ¬Fo ∧ . . .' (why?). Given the same pretence that everything has a name, that big conjunction is in effect equivalent to '∀x¬Fx'. Whence, just as we want, '∃xFx' is equivalent to the negation of that, i.e. '¬∀x¬Fx'. (Give a similar arm-waving argument for why '∀xFx' should be equivalent to '¬∃x¬Fx'.)

We could therefore do quantificational logic using a language with only one of the two interdefinable types of quantifiers just as we could do propositional logic using only one of conjunction or disjunction alongside negation. Frege's *Begriffsschrift* in fact used just the universal quantifier. However, just as it is customary and convenient to use both conjunction and disjunction in PL languages, so it is now customary and convenient to use both types of quantifier in QL languages.

(c) The whole point of the quantifier-variable notation is to enable us to represent clearly the relative scope of quantifiers, so we can render e.g. the two different readings of 'Everyone loves a certain someone' quite unambiguously by e.g the QL_1 wffs '∀x∃yLxy' and '∃y∀xLxy'. Here we plainly cannot change the order of quantifiers without changing the meaning of the wff. And that holds in general.

But there *are* exceptions that we should note. Thus consider

(1) ∀x∀yLxy

which says that of everyone what '∀yLmy' says of Socrates. So (1) says that everyone loves everyone. While, swapping round the quantifiers,

(2) ∀y∀xLxy

says that everyone has the property which '∀xLxn' attributes to Plato, i.e. the property of being loved by everyone. So (2) also says that everyone is loved by everyone. Hence (1) and (2) are equivalent, even though the quantifiers are interchanged.

Likewise

(3) ∃x∃yLxy
(4) ∃y∃xLxy

are also equivalent, both saying that someone loves someone.

The point generalizes. Suppose ξ and ζ are (distinct) variables. Then:

Adjacent quantifiers of *the same type* can be interchanged. That is to say, pairs of wffs of the form $\forall\xi\forall\zeta\varphi$ and $\forall\zeta\forall\xi\varphi$ are equivalent. And pairs of wffs of the form $\exists\xi\exists\zeta\varphi$ and $\exists\zeta\exists\xi\varphi$ are equivalent.

28.9 Summary

The atomic wffs of QL languages are built from predicates and terms; proper names are the most basic kind of term. Predicates come with fixed arities. And an atomic wff is formed by taking a predicate of arity k and following it by k terms (not necessarily different).

We give the interpretations of proper names and predicates by glossaries associating the expressions with vernacular equivalents. Then, roughly, a predicate-name(s) wff says that the named object(s) satisfy the condition expressed by the predicate.

Connectives in QL languages still basically behave as sentential connectives, as in PL languages.

We next add variables and quantifier symbols to the logical vocabulary of QL languages. Expressions like '$\forall$x', '$\forall$y', . . . , are now available as universal quantifiers: expressions like '$\exists$x', '$\exists$y', . . . , are existential quantifiers.

A wff with an initial quantifier can then be formed by taking a wff $\alpha(\tau)$ with one or more occurrences of a name or other term τ, replacing the term throughout by a variable ξ new to the wff to form $\alpha(\xi)$, and prefixing a quantifier involving the same variable to get $\forall\xi\alpha(\xi)$ or $\exists\xi\alpha(\xi)$.

We interpret quantified wffs by fixing the domain or universe of discourse for the relevant language. Suppose $\alpha(\tau)$ says the object referred to by τ is C. Then a wff of the form $\forall\xi\alpha(\xi)$ renders *everything is C*, while $\exists\xi\alpha(\xi)$ renders *something is C* – where the things in question are to be found in the relevant domain.

$\forall\xi\alpha(\xi)$ is equivalent to $\neg\exists\xi\neg\alpha(\xi)$, and $\exists\xi\alpha(\xi)$ is equivalent to $\neg\forall\xi\neg\alpha(\xi)$.

Exercises 28

(a) Which of the following expressions are wffs of the language QL_1?

(1) Mn
(2) (Lmn → Lnm)
(3) ∃x∀y∃x Rxyx
(4) ∃x(Lmx → Lnm)

29 Simple Translations

This chapter explores simple examples of translating between English and QL languages.

The striking feature of QL languages is of course that they use a *quantifier/variable* notation to express generality. However, as we have seen, this notation is in fact modelled quite closely on two familiar devices found in ordinary English and/or in mathematician's English (namely prefixed quantifiers tied to pronouns, and variables serving as pronouns). So the basics of the notation are actually not hard to grasp.

What will cause the translational headaches is the additional *sparseness* that we have imposed on our formalized languages:

(i) Each QL language has just a single sort of variable, with all variables ranging over the same objects.

(ii) QL languages don't have any special apparatus for expressing restricted quantifiers. They don't have a new kind of binary quantifier taking two predicates – notated perhaps '$\forall x(Fx : Gx)$' or '$(\forall x : Fx)\, Gx$' – to say that all Fs are G. We have to make do with the now-familiar unary quantifiers, plus connectives.

(iii) QL languages also have to manage with just two built-in kinds of quantifier.

These sparseness requirements are absolutely standard, and you have to learn to negotiate them. But we should emphasize again that they are not really instrinsic to the basic Fregean conception of quantifiers that underlies modern logic.

29.1 Restricted quantifiers revisited

(a) Suppose we are working in a language like QL_1 where the generic quantifiers range inclusively over all people, past and present. Then, given the sparse resources of this language, if we want to translate

(1) All logicians are wise,

(2) Some logicians are wise,

our only option is to follow the lead of §27.2, and restrict our quantifiers using connectives. Therefore (1) and (2) will get translated by

(3) $\forall x(Gx \to Hx)$,

(4) $\exists x(Gx \wedge Hx)$.

And let's stress that the connectives do have to go this way round – i.e. we need to translate (1) using a conditional and (2) using a conjunction, and not vice versa. Why?

(i) Translating (1) into QL_1 as '$\forall x(Gx \wedge Hx)$' would plainly be wrong. For that formal wff says, quite differently, that everyone is a wise logician!

(ii) Translating (2) as '$\exists x(Gx \rightarrow Hx)$' would also be wrong. Socrates is not a logician, and 'Gm' is false. Hence '$(Gm \rightarrow Hm)$' is *true*, being a material conditional with a false antecedent. But if a named individual satisfies a condition, then certainly something does. Correspondingly, if '$(Gm \rightarrow Hm)$' is true, so is '$\exists x(Gx \rightarrow Hx)$'. In sum, the mere existence of a non-logician like Socrates makes '$\exists x(Gx \rightarrow Hx)$' true. But obviously that's not enough to make (2) true, i.e. make it true that some logicians are wise.

Further, we want the translations of intuitively equivalent ordinary-language propositions to come out as equivalent formal sentences. 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 QL_1 translations should be equivalent.

Translating restricted 'all' with a conditional, the negated 'all' proposition gets rendered '$\neg\forall x(Gx \rightarrow \neg Hx)$', which is equivalent to '$\exists x\neg(Gx \rightarrow \neg Hx)$' (why?). But we know from propositional logic that something of the form $\neg(\alpha \rightarrow \neg\beta)$ is equivalent to $(\alpha \wedge \beta)$; and similarly, still just because of the meaning of the connectives, '$\exists x\neg(Gx \rightarrow \neg Hx)$' is equivalent to '$\exists x(Gx \wedge Hx)$'.

So putting things together, 'Not all logicians are unwise' is rendered as '$\neg\forall x(Gx \rightarrow \neg Hx)$' which comes out as equivalent to '$\exists x(Gx \wedge Hx)$', which translates 'Some logicians are wise'. As required.

(b) How good is our translation of (1) as (3)? In particular, what about the use of the material conditional here? Informally, we might suggest that

(1) All logicians are wise

can be paraphrased along the lines of

(1′) Everyone is such that, if they are a logician, they are wise.

Suppose we accept this. Then note that we have already discussed the kind of 'if' which is involved in contexts like (1′) in §19.3. Using a different example, we argued that the 'if' here does need to be a material conditional. So if we accept the equivalence of (1) and (1′), then we should indeed render both as

(3) $\forall x(Gx \rightarrow Hx)$.

(c) What about our translation of (2)? The English plural in 'Some logicians are wise' surely indicates that there is more than one wise logician. But (4) '$\exists x(Gx \wedge Hx)$' 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 *G*s are *H*' as a premiss, or aim to derive it as a conclusion, the number of *G*s that are *H* typically doesn't matter to us – we care about whether *any* of them are. So we can simply ignore the small translational infelicity in rendering 'Some *G*s are *H*' as '$\exists x(Gx \wedge Hx)$'. In almost every context it is worth the gain in formal simplicity. Further, when we later augment our QL languages with an identity predicate, we will see how to express 'some (more than one)' with the resources of our enriched language, when we really need to do so.

29.2 Existential import

(a) We have just claimed that, worries about plurals aside, propositions of the form

(1) All *G*s are *H*,
(2) Some *G*s are *H*,

can be rendered into a QL language by corresponding wffs like

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

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 wise, then there must exist some logicians to be wise. Hence a proposition of the form (1) entails the corresponding proposition of the form (2). Or so the story goes.

However, the wff (3) doesn't entail (4) (assuming it is possible for there to be no *G*s). For suppose that there *are* no *G*s. Then (3) is true: everyone x is such that (x is $G \rightarrow x$ is H) since the antecedent of the material conditional is always false. While (4) is false. Therefore (3) won'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), while (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 First 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 some particles subject to no net force: doesn't the law remain true irrespective of whether there are any such particles? 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 *G*s, while *Any Gs are H* leaves it open whether there are some *G*s – 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 or not vernacular propositions of the form *All Gs are H* usually entail the existence of some *G*s. Just regard this as one of those messy issues about ordinary language which get helpfully tidied up when we move to a formalized language.

Our policy henceforth will be to take the default rendering of *All Gs are H* (and similarly *Every/any/each G is H*) as something like '$\forall x(Gx \rightarrow Hx)$', which lacks existential import. We can then always tack an explicit existential clause '$\exists xGx$' onto the translation if, on a particular occasion, we think it matters.

29.3 'No'

The last two sections suggest that sparse QL languages can handle restricted 'all'/'every' and 'some' quantifiers reasonably well. But what about handling other quantifiers? Our QL languages have just two flavours of built-in quantifiers: should we have added e.g. a built-in 'no' quantifier?

(a) In QL_1, '$\forall x\neg Hx$' means that everyone is not wise, i.e. *no one* is wise. In the same language '$\neg\exists xHx$' 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 alternative (but equivalent) translations of that basic 'no' proposition.

The point obviously generalizes. Informally, we can recast generic 'no' propositions using a quantifier prefix like '(no one/nothing x is such that)', and then recast that as either '(everyone/everything x is such that) it is not the case that' or 'it is not the case that (someone/something x is such that)', which correspond to the formal expressions '$\forall x\neg$' or '$\neg\exists x$'.

More carefully, using the same notation as in §28.7,

> Suppose we interpret the term τ as referring to o, and suppose the wff $\alpha(\tau)$ then says that o is C. Then $\forall\xi\neg\alpha(\xi)$ and $\neg\exists\xi\alpha(\xi)$ are equivalent, and say that there are no Cs (in the relevant universe of discourse).

Of course, we *could* have added a third built-in quantifier symbol to QL languages (e.g. a – rotated! – 'N' to go with '$\forall$' and '$\exists$') so we could then express 'there are no Fs' by '$\mathsf{N}x\mathsf{F}x$'. This would make some translations rather smoother; the cost would be increasing the number of rules needed later for dealing with quantifier arguments. The conventional judgement is that the gain isn't worth the cost.

(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 goes via the Loglish

(2) (Everyone x is such that) if x is a philosopher, then x is not wise

to arrive at:

(3) $\forall x(Fx \to \neg Hx)$.

The second option formalizes the equivalent thought

(4) It is not the case that (someone x is such that) x is a philosopher, and x is wise

to get:

(5) $\neg\exists x(Fx \wedge Hx)$.

Why are (3) and (5) equivalent? $\forall x(Fx \to \neg Hx)$ is equivalent to $\neg\exists x\neg(Fx \to \neg Hx)$, and that in turn is equivalent to $\neg\exists x(Fx \wedge Hx)$, as we would expect when we recall that $\neg(\alpha \to \neg\beta)$ is equivalent to $(\alpha \wedge \beta)$.

29.4 Translating via Loglish

Faced with the task of translating from English to some QL language, we can proceed methodically going via Loglish as an intermediate step (relying on the key fact that the meanings of the QL and the Loglish quantifiers are, by design, the same.) And we can break down translation-via-Loglish into stages. So suppose we start, on the one side, with an ordinary-language proposition involving generality. And suppose we have, on the other side, a QL language with an appropriate glossary. Then:

Stage one Begin by recasting our vernacular proposition by using quantifier prefixes linked to later pronouns, translating away any 'no' quantifiers. At a first shot, these prefixes will typically be restricted quantifiers such as '(every woman is such that)', '(someone who loves Owen is such that)' or '(some prime number is such that)'.

Stage two Replace the vernacular pronouns with variables-as-pronoun 'x', 'y', etc., and make cross-linkings clear by explicitly tagging each quantifier prefix with the variable it is linked to, to get the likes of '(every woman x is such that)', '(someone y who loves Owen is such that)', '(some prime number z is such that)'.

Stage three The glossary for our formal language has fixed a domain for its quantifiers to range over. So now replace the informal restricted quantifiers by using informal unary quantifiers over this domain plus conditionals/conjunctions. So, for example, '(every woman x is such that)' becomes '(everyone x is such that) if x is a woman, then . . . ', '(someone y who loves Owen is such that)' becomes '(someone y is such that) y loves Owen and . . . ', etc.

Stage four Next, simply abbreviate the generic quantifier prefixes by '$(\forall x)$', '$(\exists y)$'. Also replace the vernacular connectives with their symbolic versions, and we get Loglish sentences like:

$(\forall x)(x \text{ is a woman} \rightarrow x \text{ is an adult human})$,
$(\exists y)(y \text{ loves Owen} \wedge y \text{ is a philosopher})$,
$(\forall z)(z \text{ is a philosopher} \rightarrow (\exists y)(y \text{ is a logician} \wedge x \text{ admires } y))$.

These are *still* quite straightforwardly interpretable, relying on our understanding of the symbols and our grasp of English.

Stage five Finally, replace the vernacular names and predicates in Loglish sentences with symbolic counterparts using our chosen QL language's glossary, making sure that we now obey the predicates-first 'word order'. Rewrite '$(\forall x)$', '$(\exists y)$' in the formal style, without brackets, as '$\forall$x', '$\exists$y'.

Of course, it is not being seriously suggested that you have to follow this stage-by-stage plan in all its detail and in the exact order suggested! You will very quickly learn to skip lightly through these stages when translating from English to QL.

Still, in our illustrative examples in the next section it will be helpful to take things very slowly and show our working, going for rather plodding explicitness with the aim of maximum clarity.

29.5 Translations into QL_2

(a) Nothing would be gained at this point by taking sober mathematical examples (for instance), and we could run the risk of making things look more difficult than they really are. So let's stick to mundane human affairs and define the following language:

In QL_2, the proper names with their interpretations are

m: Maldwyn,
n: Nerys,
o: Owen;

and the predicates are

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.

The domain of quantification: people (living people, for definiteness).

And now let's consider how to translate the following bunch of propositions into QL_2:

(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) Maldwyn is married to no one.
(8) No one who loves Nerys loves Owen.
(9) Nerys is a woman everyone loves.
(10) If everyone loves Nerys then Owen does.
(11) If someone loves Nerys then Owen does.
(12) If anyone loves Nerys then Owen does.

Try your hand at these examples before reading on!

(b) Taking our propositions in turn, we have first:

(1) There's someone who loves Maldwyn, Nerys and Owen
≃ (Someone is such that) they love Maldwyn, Nerys and Owen
≃ (Someone x is such that) x loves Maldwyn, Nerys and Owen
≃ $(\exists x)(x$ loves Maldwyn $\wedge$ x loves Nerys $\wedge$ x loves Owen)
≃ $\exists x(\mathsf{Lxm} \wedge (\mathsf{Lxn} \wedge \mathsf{Lxo}))$.

For brevity, we will use the sign '≃' to indicate sufficient translational equivalence, as far as truth-relevant meaning is concerned. 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'. (In other examples too, you can use any

variables you like, so long as you preserve the crucial pattern of linkages from quantifier prefixes to slots in later expressions.)

(2) Whoever is loved by Owen is loved by Maldwyn too
- ≃ (Everyone who is loved by Owen is such that) Maldwyn loves them
- ≃ (Everyone x who is loved by Owen is such that) Maldwyn loves x
- ≃ (Everyone x is such that)(if x is loved by Owen then Maldwyn loves x)
- ≃ $(\forall x)$(Owen loves $x \rightarrow$ Maldwyn loves x)
- ≃ $\forall x(\mathsf{Lox} \rightarrow \mathsf{Lmx})$.

Note how the relative clause 'who is loved by Owen' is treated as a predicate restricting the quantifier.

(3) Every woman who loves Maldwyn is loved by Owen
- ≃ (Every woman x who loves Maldwyn is such that) x is loved by Owen
- ≃ (Everyone x is such that)(if x is a woman who loves Maldwyn, then Owen loves x)
- ≃ $(\forall x)(x$ is a woman and x loves Maldwyn $\rightarrow$ Owen loves $x)$
- ≃ $\forall x((\mathsf{Gx} \wedge \mathsf{Lxm}) \rightarrow \mathsf{Lox})$.

(4) Maldwyn loves some woman who loves Owen
- ≃ (Some woman x who loves Owen is such that) Maldwyn loves x
- ≃ (Someone x is such that)(x is a woman who loves Owen, and Maldwyn loves x)
- ≃ $(\exists x)(x$ is a woman and x loves Owen $\wedge$ Maldwyn loves $x)$
- ≃ $\exists x((\mathsf{Gx} \wedge \mathsf{Lxo}) \wedge \mathsf{Lmx})$.

(5) Nerys and Owen love any child of theirs
- ≃ (Everyone x who is a child of Nerys and Owen is such that) Nerys and Owen love x
- ≃ (Everyone x is such that)(if x is a child of Nerys and Owen then Nerys and Owen love x)
- ≃ $(\forall x)(x$ is a child of Nerys and Owen $\rightarrow$ (Nerys loves $x \wedge$ Owen loves $x))$
- ≃ $\forall x(\mathsf{Rxno} \rightarrow (\mathsf{Lnx} \wedge \mathsf{Lox}))$.

(6) Owen is Maldwyn's child
- ≃ Owen is a child of Maldwyn
- ≃ (Someone x is such that) Owen is a child of Maldwyn and x
- ≃ $\exists x\, \mathsf{Romx}$.

The point of this last example is to emphasize that QL_2's predicate 'R' is determinately a *ternary* predicate, glossed as '① is a child of ② and ③'. If we want to translate 'Owen is a child of Maldywn' which involves the ordinary-language *binary* predicate '① is a child of ②', we have to use the illustrated trick of sopping up a place in the ternary predicate with a quantified variable.

Now for a couple of simple examples involving the 'no' quantifier. Remember, §29.3 tells us that that 'no' propositions can be rendered in two different ways. Thus:

(7) Maldwyn is married to no one
- ≃ It is not the case that Maldwyn is married to someone

≃ ¬(there is someone x such that) Maldwyn is married to x
≃ $\neg\exists x\mathsf{M}\mathsf{m}x$.

Or, starting again, taking the alternative route:

≃ (Everyone x is such that) Maldwyn isn't married to x
≃ $\forall x\neg\mathsf{M}\mathsf{m}x$.

(8) No one who loves Nerys loves Owen
≃ It is not the case that someone who loves Nerys loves Owen
≃ ¬(there is someone x who loves Nerys and is such that) x loves Owen
≃ ¬(there is someone x who is such that)(x loves Nerys and x loves Owen)
≃ $\neg\exists x(\mathsf{L}x\mathsf{n} \wedge \mathsf{L}x\mathsf{o})$.

Or equivalently

≃ 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)(if x loves Nerys then x doesn't love Owen)
≃ $\forall x(\mathsf{L}x\mathsf{n} \rightarrow \neg\mathsf{L}x\mathsf{o})$.

The next proposition can also be regimented in two related ways:

(9) Nerys is a woman everyone loves
≃ Everyone loves Nerys and Nerys is a woman
≃ $(\forall x\mathsf{L}x\mathsf{n} \wedge \mathsf{G}\mathsf{n})$.

Or alternatively

≃ (Everyone is such that) they love Nerys-who-is-a-woman
≃ $\forall x(\mathsf{L}x\mathsf{n} \wedge \mathsf{G}\mathsf{n})$.

We will say more about why these wffs are equivalent in the next section.

Next, a conditional with complete propositions as antecedent and consequent:

(10) If everyone loves Nerys then Owen does
≃ ((everyone x is such that) x loves Nerys → Owen loves Nerys)
≃ $(\forall x\mathsf{L}x\mathsf{n} \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$.

Similarly,

(11) If someone loves Nerys then Owen does
≃ $(\exists x\mathsf{L}x\mathsf{n} \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$.

But now what about (12) 'If anyone loves Nerys then Owen does'? Perhaps with the right stress and/or right context we can construe this as saying 'If just *anyone* loves Nerys, then . . . ' which is equivalent to (10). However, the more natural reading gives 'anyone' wider scope than the conditional – as we noted before, in §26.2 and §26.3(b), 'every' and 'any' typically behave differently in the antecedents of conditionals. So on this reading,

(12) If anyone loves Nerys then Owen does
≃ (Anyone/everyone is such that) if they love Nerys, then Owen loves Nerys
≃ $\forall x(\mathsf{L}x\mathsf{n} \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$.

However, can't we can equally naturally read (12) as in fact equivalent to (11) and translate it accordingly? Well, yes we can: the alternative translations '$\forall x(\mathsf{L}x\mathsf{n} \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$' and '$(\exists x\mathsf{L}x\mathsf{n} \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$' are in fact equivalent – as we will see in the next section.

29.6 Moving quantifiers

(a) Consider again (9) from the previous section, i.e. 'Nerys is a woman everyone loves' and its two renditions:

$(\forall x Lxn \wedge Gn)$
$\forall x(Lxn \wedge Gn)$.

These wffs are indeed equivalent given our account of how to interpret the notation. (If you don't see this immediately, it might help to pretend there are just the three named people in the domain and then note that – when properly bracketed –

$(Lmn \wedge Lnn \wedge Lon) \wedge Gn$
$(Lmn \wedge Gn) \wedge (Lnn \wedge Gn) \wedge (Lon \wedge Gn)$

are equivalent.)

The point generalizes. Pairs of wffs of the following forms are equivalent:

$(\forall \xi \alpha(\xi) \wedge \beta)$ and $\forall \xi(\alpha(\xi) \wedge \beta)$.

Or at least, this is true so long as the wff β doesn't contain the variable ξ. That restriction is crucial. For example, in one direction, we can't go from the wff '$(\forall x Fx \wedge \forall x Gx)$' to '$\forall x(Fx \wedge \forall x Gx)$' – the latter expression isn't even a wff in our syntax. Similarly, we can't go in the other direction from the wff '$\forall x(Fx \wedge Gx)$' to '$(\forall x Fx \wedge Gx)$' – again the latter expression isn't a wff.

Similar equivalences hold for the existential quantifier, and for disjunction in place of conjunction. So we have in summary:

> Pairs of wffs of the following forms are equivalent:
>
> $(\forall \xi \alpha(\xi) \wedge \beta)$ and $\forall \xi(\alpha(\xi) \wedge \beta)$
> $(\exists \xi \alpha(\xi) \wedge \beta)$ and $\exists \xi(\alpha(\xi) \wedge \beta)$
> $(\forall \xi \alpha(\xi) \vee \beta)$ and $\forall \xi(\alpha(\xi) \vee \beta)$
> $(\exists \xi \alpha(\xi) \vee \beta)$ and $\exists \xi(\alpha(\xi) \vee \beta)$
>
> where β doesn't contain the variable ξ. The equivalencies also hold with the order of the conjunctions/disjunctions reversed.

(b) What about moving a quantifier outside a conditional? This is more interesting.

$(\exists x Lxn \rightarrow Gn)$

is equivalent, by the definition of the material conditional, to

$(\neg \exists x Lxn \vee Gn)$

which by the interrelation of the quantifiers is equivalent to

$(\forall x \neg Lxn \vee Gn)$.

But by the rule for taking a universal quantifier outside a disjunction, this is in turn equivalent to

$\forall x(\neg Lxn \vee Gn)$

which is equivalent to

$\forall x(Lxn \rightarrow Gn)$

by the interpretation of the material conditional again. Which gives us the claimed overall equivalence between the renditions of example (12) in §29.5. Dragging the quantifier out from the *antecedent* of the conditional flips the existential quantifier into a universal.

For similar reasons, these for example are equivalent:

$(\forall xLxn \rightarrow Gn)$
$\exists x(Lxn \rightarrow Gn)$.

However, since the consequent of a material conditional is like an ordinary, unnegated, disjunct we can pull out a quantifier as with other disjunctions. Generalizing, then:

Pairs of wffs of the following forms are equivalent:

$(\forall\xi\alpha(\xi) \rightarrow \beta)$ and $\exists\xi(\alpha(\xi) \rightarrow \beta)$
$(\exists\xi\alpha(\xi) \rightarrow \beta)$ and $\forall\xi(\alpha(\xi) \rightarrow \beta)$
$(\beta \rightarrow \forall\xi\alpha(\xi))$ and $\forall\xi(\beta \rightarrow \alpha(\xi))$
$(\beta \rightarrow \exists\xi\alpha(\xi))$ and $\exists\xi(\alpha(\xi) \rightarrow \beta)$

where β doesn't contain the variable ξ.

You do have to keep your wits about you when simultaneously handling conditionals and moving quantifiers around!

29.7 Summary

The use of QL languages to regiment quantified propositions for logical purposes really comes into its own when we are dealing with more complex multiply-quantified propositions. We'll look at examples of these in the next Chapter. But let's pause to take stock so far.

Restricted universal quantifiers are translated using conditionals in the scope of universal quantifiers which run over the whole domain. Restricted existential quantifiers are translated using conjunctions in the scope of existential quantifiers which run over the whole domain.

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 *G*s' by the corresponding wff '$\neg\exists xGx$' or equivalently '$\forall x\neg Gx$'. And the restricted quantification 'No *F*s are *G*s' can be rendered by either '$\neg\exists x(Fx \wedge Gx)$' or '$\forall x(Fx \rightarrow \neg Gx)$'.

We can ease the process of translating from English into a given QL language by going in stages via Loglish:

(1) Rephrase a given English proposition using prefixed restricted quantifiers linked to pronouns (massaging away 'no' quantifiers using 'every' or 'some' plus negation).
(2) Replace vernacular pronouns with variables, and make cross-linkings clear by tagging quantifier prefixes with the variables they are linked to.

(3) Render the restricted quantifiers by using generic quantifiers (running over the whole universe) together with conditionals and conjunctions,
(4) Use '∀' and '∃' to abbreviate quantifier prefixes, replace vernacular connectives with formal ones, giving us Loglish expressions.
(5) Then use the glossary for the relevant QL language to render the names and predicates.

Exercises 29

30 More on translations

This chapter starts by looking at more difficult examples of translating between English and the particular formal language QL_2. Then we consider some issues about how to choose a QL language to use when assessing arguments.

30.1 More translations into QL_2

(a) Let's take some more examples of translating claims about Welsh affairs, this time ones which involve multiple quantifiers. So let's consider:

(1) Every man loves someone or other.
(2) Some man loves a woman.
(3) Someone Owen loves is loved by everyone Nerys loves.
(4) No woman loves every man.
(5) No woman loves any man.
(6) Anyone who is married loves someone they aren't married to.
(7) A married man only loves women.
(8) No man loves anyone who loves Nerys.

A helpful tip: In going via Loglish intermediate stages, it will be useful if we occasionally make informal use of square brackets to demarcate parts of complex expressions; then we can work within the square brackets as we move from one stage to the next. So:

(1) Every man loves someone or other
  - $\simeq$ (Every man x is such that) [x loves someone or other]
  - $\simeq$ (Everyone x is such that) if x is a man, then [x loves someone or other]
  - $\simeq$ $(\forall x)(x$ is a man $\rightarrow [(\exists y)\ x$ loves $y])$
  - $\simeq$ $\forall \mathsf{x}(\mathsf{Fx} \rightarrow \exists \mathsf{y}\mathsf{Lxy})$.

(2) Some man loves a woman
  - $\simeq$ (There is some man x such that)[(there is some woman y such that) x loves y]
  - $\simeq$ (There is someone x such that)(x is a man and [(there is some woman y such that) x loves y])
  - $\simeq$ $(\exists x)(x$ is a man $\wedge$ [(there is someone y such that) y is a woman and x loves y])
  - $\simeq$ $(\exists x)(x$ is a man $\wedge [(\exists y)(y$ is a woman $\wedge\ x$ loves $y)])$
  - $\simeq$ $\exists \mathsf{x}(\mathsf{Fx} \wedge \exists \mathsf{y}(\mathsf{Gy} \wedge \mathsf{Lxy}))$.

An equivalent translation would be '$\exists x \exists y((Fx \wedge Gy) \wedge Lxy)$' (why?).

(3) Someone Owen loves is loved by everyone Nerys loves
- $\simeq$ (Someone x whom Owen loves is such that)[x is loved by everyone Nerys loves]
- $\simeq$ (Someone x is such that)(Owen loves x and [x is loved by everyone Nerys loves])
- $\simeq$ (Someone x is such that)(Owen loves x and [(everyone y is such that) if Nerys loves y then y loves x])
- $\simeq$ $(\exists x)$(Owen loves $x \wedge [(\forall y)$(Nerys loves $y \rightarrow y$ loves $x)]$)
- $\simeq$ $\exists x(Lox \wedge \forall y(Lny \rightarrow Lyx))$.

(4) No woman loves every man
- $\simeq$ It is not the case that some woman loves every man
- $\simeq$ $\neg$ Some woman x is such that x loves every man
- $\simeq$ $\neg$(Someone x is such that)(x is a woman and [x loves every man]
- $\simeq$ $\neg$(Someone x is such that)(x is a woman and [(every y is such that) if y is a man, then x loves y]
- $\simeq$ $\neg \exists x(Gx \wedge \forall y(Fy \rightarrow Lxy))$.

Or, starting again, taking the alternative route for translating a 'no' proposition:
- $\simeq$ Every woman is such that she doesn't love every man
- $\simeq$ (Every woman x is such that) it is not the case that x loves every man
- $\simeq$ (Every x is such that)(if x is a woman, then $\neg$[x loves every man])
- $\simeq$ (Every x is such that)(x is a woman $\rightarrow \neg$[(everyone y is such that) if y is a man, x loves y])
- $\simeq$ $\forall y(Gx \rightarrow \neg \forall y(Fy \rightarrow Lxy))$.

Next, the stressed claim 'No woman loves just *any* man' is naturally heard as saying the same as (4). But an unstressed (5) 'No woman loves any man' is more naturally read as we might also read 'No woman loves a man' (compare 'No five year old reads any volumes of Proust'). Taking this reading, rendering 'no' as 'not some', and then following the pattern in our translation of (2), we have

(5) No woman loves any man
- $\simeq$ It is not the case that [some woman loves a man]
- $\simeq$ $\neg \exists x(Gx \wedge \exists y(Fy \wedge Lxy))$.

An equally good rendition, using our other style for translating 'no' propositions, is

- $\simeq$ $\forall x(Gx \rightarrow \neg \exists y(Fy \wedge Lxy))$.

Here's a third:

No woman loves any man
- $\simeq$ Every woman is such that every man is such that she doesn't love him
- $\simeq$ (Everyone x is such that)(if x is a woman then [every man y is such that x doesn't love y])
- $\simeq$ (Everyone x is such that)(if x is a woman then [(everyone y is such that) if y is a man, then x doesn't love y])
- $\simeq$ $\forall x(Gx \rightarrow \forall y(Fy \rightarrow \neg Lxy))$.

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 the unary 'is married' to 'is married to someone'. Thus we have:

(6) Anyone who is married loves someone they aren't married to
- ≃ (Everyone x who is married is such that)[there is someone y whom x isn't married to, such that x loves y]
- ≃ (Everyone x is such that)(if x is married, then [(someone y is such that) x isn't married to y and x loves y])
- ≃ $\forall x(x \text{ is married} \rightarrow \exists y(\neg Mxy \wedge Lxy))$.

Finally, we render 'x is married' using 'M' plus an existential quantifier. What variable shall we use? Rule of thumb: when you need to introduce some variable into a wff, it is always safer to use one that doesn't already appear. That way you will avoid unintended tangles. So, choose 'z', and we can finish with

- ≃ $\forall x(\exists z Mxz \rightarrow \exists y(\neg Mxy \wedge Lxy))$.

In our next example on the list, 'A married man . . . ' is naturally read as a universal generalization about married men. So, we can begin

(7) A married man only loves women
- ≃ (Every man x who is married is such that)(x only loves women)
- ≃ (Everyone x is such that)(if x is a man and is married, then [x only loves 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 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 y then y is a woman])
- ≃ $\forall x((Fx \wedge \exists z Mxz) \rightarrow \forall y(Lxy \rightarrow Gy))$.

Our final example again involves that troublesome vernacular quantifier 'anyone'. As we have already seen, in some contexts, 'anyone' is replaceable by 'someone'; (8) seems to be another case in point. The following rendition seems natural:

(8) No man loves anyone who loves Nerys.
- ≃ (Every man x is such that) it is not the case that [there is someone who loves Nerys whom x loves]
- ≃ (Every man x is such that) ¬[(there is someone y such that) y loves Nerys ∧ x loves y]
- ≃ $(\forall x)(x \text{ is a man} \rightarrow \neg[(\exists y)(y \text{ loves Nerys} \wedge x \text{ loves } y)])$
- ≃ $\forall x(Fx \rightarrow \neg\exists y(Lyn \wedge Lxy))$.

An alternative translation could be '$\forall x(Fx \rightarrow (\forall y)(Lyn \rightarrow \neg Lxy))$'. Why should these be equivalent?

And that's more than enough examples to be going on with!

30.2 Translations from QL_2

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/or '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 just reverse the stage-by-stage-via-Loglish procedure that we have just been using.

To illustrate this, we will take three QL_2 wffs:

(1) $\neg\forall x\forall y((Fx \wedge Gy) \rightarrow (Mxy \rightarrow \exists z Rzxy))$
(2) $\forall x(Gx \rightarrow \forall y\forall z(Rxyz \rightarrow (Lxy \wedge Lxz)))$
(3) $(\forall x(Gx \rightarrow \neg Lxm) \rightarrow \forall x((Lxm \rightarrow \exists y Rxmy) \wedge (\exists y Rxmy \rightarrow Lxm)))$

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

(1) $\neg\forall x\forall y((Fx \wedge Gy) \rightarrow (Mxy \rightarrow \exists z Rzxy))$
  - $\simeq$ ¬(For anyone x and anyone y)(if x is a man and y a woman, then if x is married to y, there is someone z who is x and y's child)
  - $\simeq$ ¬(For any man and any woman)(if they are married to each other, they have a child together)
  - $\simeq$ Not every man and woman who are married have a child together.

(2) $\forall x(Gx \rightarrow \forall y\forall z(Rxyz \rightarrow (Lxy \wedge Lxz)))$
  - $\simeq$ (Everyone x is such that)(if x is a woman then [everyone y and everyone z are such that (if x is a child of y and z, then x loves y and x loves z)]
  - $\simeq$ Every woman x is such that [for anyone y and anyone z (if x is a child of y and z then x loves y and x loves z)]
  - $\simeq$ Every woman loves her parents.

(3) $(\forall x(Gx \rightarrow \neg Lxm) \rightarrow \forall x((Lxm \rightarrow \exists y Rxmy) \wedge (\exists y Rxmy \rightarrow Lxm)))$
  - $\simeq$ If $\forall x(Gx \rightarrow \neg Lxm)$, then $\forall x((Lxm \rightarrow \exists y Rxmy) \wedge (\exists y Rxmy \rightarrow Lxm))$
  - $\simeq$ If no woman loves Maldwyn, then $\forall x(Lxm$ if and only if $\exists y Rxmy)$
  - $\simeq$ If no woman loves Maldwyn then, for any x, x loves Maldwyn if and only if x is a child of Maldwyn and someone
  - $\simeq$ If no woman loves Maldwyn, then he is loved just by any children he has.

30.3 Choosing a QL language

Some QL languages are of stable and lasting interest. Two examples which we will meet later: the language of set theory (see §**??**), and the language of first-order arithmetic (see §**??**). But most of the languages which we meet in this book are of much more fleeting interest. They are constructed ad hoc, for temporary use, either just for illustrative purposes (like QL_1 and QL_2), or when we want to formalize particular arguments (starting in Chapter 32). So, when rendering ordinary language into some ad hoc QL, how do we choose a language? More specifically, how do we choose which domain to quantify over?

Suppose, for example, that we want to regiment the argument

(1) Some philosophers are wise; no astrologer is wise; so some philosophers are not astrologers.

Ignoring the implication that there is more than one wise philosopher, we can regiment the argument into an ad hoc QL language along the lines of

(2) $\exists x(Px \wedge Wx),\ \forall x(Ax \rightarrow \neg Wx) \ \therefore\ \exists x(Px \wedge \neg Ax)$,

where the predicate letters get the obvious interpretation and the domain of quantification is naturally chosen to be people.

But what if we want to regiment the equally valid argument

(3) Some philosophers are wise; no koala bear is wise; so some philosophers are not koala bears.

This time, we can't use a formal language whose quantifiers run just over people. We need a language with a more inclusive domain, one that includes both philosophers and koalas. Then we can formalize the argument like this:

(4) $\exists x(Px \wedge Wx),\ \forall x(Kx \rightarrow \neg Wx) \ \therefore\ \exists x(Px \wedge \neg Kx)$,

where the quantifier runs over e.g. mammals, or living things, or terrestrial objects, or over any other sufficiently inclusive domain. And we could indeed have used a more inclusive domain in regimenting (1) too.

So, the same premiss 'Some philosophers are wise' can be equally well translated into different QL languages with different domains. And which language we choose can depend on what else we want to be able to translate into the same language. Generalizing:

> When choosing a QL language for regimenting some given ordinary-language proposition(s), we must pick a language whose domain of quantification includes everything that the informal quantifiers in the given propositions are already quantifying over. But there will be general be no single correct choice.

30.4 'Translation' and 'logical form'

After busily grappling with so many examples of translation to and from QL languages over the last two chapters, let's finish with some more restful historical/philosophical reflections!

(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 in some way 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?

(b) Here's an obvious initial problem. As we have seen, if we try to render e.g. 'No woman loves Maldwyn' into a language like QL_2, we arrive at the alternatives '$\neg\exists x(Gx \wedge Lxm)$' and '$\forall x(Gx \rightarrow \neg Lxm)$'. 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. So 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 imposed on ourselves. To keep our formal language sparse, we have stipulated that there isn't a 'no' quantifier built into the language, and also that the remaining quantifiers all range over a single universe of discourse with restrictions to be made by using connectives. So the particular shapes of our two formal versions of 'No woman loves Maldwyn' are really an artefact of these two optional choices we made for the sake of convenience in designing our QL languages; it would be hard to argue that the results of *these* choices should be deeply revealing about the semantic structure of ordinary language.

(c) Still, what about the most fundamental thing, the basic quantifier/variable idea for representing general claims? Frege and his followers are now surely onto something.

For we can agree that 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. For grammatically speaking, the two sorts of expression very often behave in much the same way – e.g. we can plug either kind into predicates like '① is Welsh' or '① loves ②' and get grammatical sentences.

Now, it would be a step too far to say that ordinary-language expressions 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 'Maldwyn wicked this way comes', or contrast 'Someone brave rescued the dog' with 'Nerys brave rescued the dog'. Or again, 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.

In particular, everyday language does not mark the crucial fact that proper names are scopeless while quantifiers have scopes, and hence can generate scope-ambiguities. So, yes, it might well be said that the quantifier/variable notation represents more perspicuously aspects of underlying semantic structure which are somewhat masked by the surface grammar of English.

(d) But we need not really press that last point. For our key claim is that it is a good strategy for logical purposes to sidestep the messy complexities of English usage – think again of 'any' vs 'every', think again of scope phenomena, etc. And we do this by (1) translating arguments involving general claims into a nicely behaved, unambiguous QL language, and then (2) evaluating the arguments in their tidily regimented form. To

defend this strategy, it is not necessary to suppose that the artificial languages (even in part) reflect structures which are in some good sense 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, and then deal with the clean-and-tidy formalized arguments – we would ideally both like (i) very direct, natural, unforced translations into the relevant artificial language L, and also like (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 distinctions we may find ourselves needing 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 PL, 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 sparse 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 connectives.

But arguably the price is right: the formal translations, although sometimes strained and cumbersome, do enable us to capture the logically-relevant content of a wide class of ordinary claims. In particular, we can then 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.

30.5 Summary

We have looked at examples of the translation of more complex, multiply quantified, propositions into QL, again easing the journey by going via Loglish intermediate stages.

We have also seen a few examples of the reverse journey, taking us back from QL wffs to their English translations.

In fixing on a formal language for rendering a given argument, we will need – among things – to assign it a sufficiently inclusive (non-empty) domain of quantification. There need be no unique way of doing this.

We do *not* claim that our renditions of e.g. restricted quantifiers reveal the underlying logical form of our vernacular propositions – so that 'All *F*s 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.

Exercises 30

Interlude: Arguing in QL

(a) We said in the previous Interlude that we will adopt a divide and rule strategy for coping with arguments involving quantifiers. We will sidestep the messy complexities of ordinary language by first regimenting arguments into well-behaved formal languages, and then assessing the resulting formalized arguments. And we have already said enough about QL languages to be able to start discussing how to do this for a wide class of quantifier arguments.

Recall again that we approached questions of PL validity in two ways:

(P1) We defined tautological validity for arguments involving the PL connectives. And we gave a direct brute-force method for determining validity in this sense.

(P2) We set down some intuitively correct truth-preserving modes of inference for the connectives, guided by their intuitive meaning, and codified these rules into a natural deduction system for warranting inferences.

Soundness and completeness theorems tell us that two approaches end up validating just the same inferences.

Similarly we can approach questions of QL validity in two ways:

(Q1) We can define a suitable notion of *q-validity* for arguments involving the quantifiers (as well as the connectives).

(Q2) We can set down some intuitively correct truth-preserving modes of inference for the quantifiers to add to the rules for the connectives, and codify them into a natural deduction system for warranting QL inferences.

The new rules that we will need to extend our Fitch-style natural deduction system to cope with quantificational reasoning turn out to be surprisingly straightforward: we will meet them over the next two chapters. But let's pause to say just something here about the new idea of q-validity.

(b) Recall how things went for propositional logic. The PL logical operators are truth-functional. This means that a *valuation* of some atomic wffs will fix the values of all the wffs constructed from these atoms. And this motivates the definition of tautological validity. A PL inference is tautologically valid if, on every possible valuation of its non-logical vocabulary (the relevant atoms), if the premisses are true then the conclusion is true too.

Similarly, the QL logical operators – now including the quantifiers – might be said to be value-functional. What this means is that fixing the truth-relevant values of some names and some predicates, plus fixing the domain of the quantifiers, will fix the values of

all the wffs we can construct from those resources. And what are the truth-determining values of names and predicates? What are their *q-values* for short? Names will get assigned references and predicates get assigned extensions (exactly as we'd expect from our preliminary discussions in Chapter 25).

This motivates the definition of q-validity. A QL inference is q-valid if, on every possible q-valuation of the relevant language – i.e. every way of assigning q-values to its names and predicates, and every way of fixing the domain – if the inference's premisses are true then the conclusion is true too.

Take the following everyday argument:

> Felix is a cat. All cats are scary. So, Felix is scary.

Obviously this is deductively valid – and moreover, it is intuitively *logically* valid, because it is valid in virtue of the meaning of the topic-neutral quantifier 'all'. Rendered into a suitable QL language, the argument becomes

$$Fn,\ \forall x(Fx \rightarrow Gx)\ \therefore\ Gn.$$

And this is q-valid. It doesn't matter what domain the quantifiers are ranging over, or which object in the domain 'n' picks out, or which sets of objects from the domain 'F' and 'G' have as extensions: on *any* such q-valuation, if the premisses are true, the conclusion is true too. (Or so we claim: think through why this really should be true!)

(c) However, even with what little we have said so far, you can immediately spot that there is going to be a fundamental disanalogy between assessing the tautological validity of PL inferences and assessing the q-validity of QL inferences.

In the PL case, the non-logical vocabulary of an inference comprises just the relevant atomic wffs; and each of these is assigned one of two truth-values. Given a finite handful of atoms, there is only a finite number of different possible assignments of truth-relevant values to them. *That* is why we can do a brute-force truth-table test to decide questions of tautological validity. There is only a limited number of possibilities to consider: so when we trudge through all the different valuations of the atoms occurring in an inference, in order to see whether a 'bad line' with true premisses and false conclusion turns up, we know the process must terminate with a verdict.

In the QL case, the non-logical vocabulary of an inference comprises (say) some names and predicates; and the values we assign these are, respectively, objects on the one hand and extensions on the other. But now note that, even in the simple 'Felix' example with its one name and two predicates, there are *countless* different values we can potentially assign to the relevant vocabulary – countless different choices of domain, choices of references for the name, and extensions for the predicates. So there is plainly no possible way of doing a brute-force search through all these q-valuations. There is in general no equivalent of doing a truth-table test to decide the q-validity of QL arguments. (There are tricks we can use for some simple cases, but there is no generally applicable test.)

(d) So what other approaches can we use to demonstrate the q-validity of logical valid QL arguments? One possibility is to develop the truth-tree method for propositional logic which we gestured at so very briefly in §16.2 (and explained in an Appendix). But in this book we continue to go down the natural deduction path.

We now know how to establish the tautological validity of PL arguments using Fitch-style proofs. And, as we promised, this framework can be smoothly and naturally(!) extended to give us proofs of the validity – indeed, the q-validity – of logical valid QL inferences too. In fact things go *so* smoothly that it is very inviting to look at proofs *first*, before saying more about q-validity.

Therefore, starting in the next chapter, we will begin to explore a natural deduction system for quantificational arguments expressed in QL languages (or rather, in QL languages as so far developed: there's more to be said about such languages, but we are putting that on hold). At the outset, we just claim that our rules of inference are intuitively compelling, capturing everyday modes of inference. Only once we have explored the formal proof system for a while will we return to give an official account of q-validity and explain why our proof-building rules are indeed reliable for q-validity.

31 Informal quantifier rules

This chapter briskly reviews four intuitively correct principles for arguing *informally* with 'every' and 'some' (regimented as quantifier prefixes).

31.1 Arguing with universal quantifiers

(a) Let's begin with that hackneyed old favourite

A Socrates is a man. All men are mortal. Hence Socrates is mortal.

This hardly needs a proof to warrant it! But consider this argument, where we render the second premiss using a quantifier-variable notation, with the quantifier prefix running over a suitably inclusive domain:

(1) Socrates is a man. (premiss)
(2) (Everything x is such that) if x is a man, then x is mortal. (premiss)
(3) If Socrates is a man, then Socrates is mortal. (from 2)
(4) Socrates is mortal. (from 1, 3)

Here the principle that is invoked at line (3) is the obvious one: what applies to everything/everyone in some relevant domain applies to any individual thing in the domain. As we can put it (using the notion of an 'instance' introduced in §27.5): *from a universal generalization, we can infer any particular instance*. That's easy!

To introduce our second principle for arguing with prefixed universal generalizations – which takes just a bit more care to explain – take this next argument:

B Everyone likes pizza. Whoever likes pizza likes ice cream. So everyone likes ice cream.

Again, this is obviously valid. But how could we derive the conclusion from the premisses with an informal proof? Well, consider:

(1) (Everyone x is such that) x likes pizza. (premiss)
(2) (Everyone x is such that) if x likes pizza, then x likes ice cream. (premiss)
Now pick any person in the domain, temporarily dub them 'Alex'. Then:
(3) Alex likes pizza (from 1)
(4) If Alex likes pizza, Alex likes ice cream (from 2)
(5) Alex likes ice cream. (from 3, 4)
But Alex was arbitrarily chosen, and we appeal to no special facts about them: so what we can deduce about them applies to everyone:
(6) (Everyone x is such that) x likes ice cream. (from 5)

The final step here is not, repeat *not*, relying on the hopeless idea that whatever is true of some individual in a domain is true of everyone/everything. Here's one person: Veronika is – as it happens – a woman, Slovak, and plays the violin. Plainly, we can't infer that everyone is a female Slovak violinist.

To repeat, the principle at stake is as follows. *Suppose we can show that an arbitrary representative member of a domain is F; then we can infer that everything in the domain is F* (where the conclusion really is general, i.e. no longer mentions the arbitrary representative). But when can we treat some individual as an arbitrary representative? When *we rely on no special distinguishing facts about that individual*. In other words, when we make no special assumptions about that individual – so then we can only draw on general knowledge about the domain in establishing that the thing in question is *F*.

Let's see this new principle of universal generalization at work again. So take an Aristotelian syllogism (compare Exercises **??**):

C No horses are green. All frogs are green. So no horses are frogs.

Regimenting the premisses and conclusion, recasting the 'no' propositions using prefixed universal quantifiers and negation, we can argue as follows:

(1) (Everything x is such that) if x is a horse, x is not green. (premiss)
(2) (Everything x is such that) if x is a frog, then x is green. (premiss)
Now pick any thing in the domain, temporarily dub it 'Arb'. Then:
(3) If Arb is a horse, Arb is not green (from 1)
(4) If Arb is a frog, Arb is green (from 2)
(5) If Arb is a horse, Arb is not a frog (from 3, 4)
But Arb was arbitrarily chosen, and we appeal to no special facts about it: so what we can deduce about it applies equally to anything.
(6) (Everything x is such that) if x is a horse, x is not a frog. (from 5)

We derive (3) and (4) using our first quantifier principle. We derive (6) using our second principle. The intermediate step from (3) and (4) to (5) is just familiar propositional reasoning.

31.2 Arguing with existential quantifiers

(a) Now we introduce two principles for arguing with 'some'. Consider first the trivial inference:

D Socrates is a wise man. Therefore some man is wise.

Regimenting the argument with a quantifier prefix, we can write simply

(1) Socrates is a man and Socrates is wise. (premiss)
(2) (Something x is such that) x is a man and x is wise. (from 1)

The principle here – call it existential generalization – is again easy: what is true of any particular named individual is certainly true of *something* in a suitably inclusive domain. Or we can put it this way: (1) can be said to be an instance of the existentially quantified proposition (2): and the principle is that *we can infer an existential generalization from any particular instance of it*.

It is worth noting too this similar argument:

E Narcissus loves himself. Therefore someone loves Narcissus.

Again, a regimented proof is simple:

(1) Narcissus loves Narcissus. (premiss)
(2) (Someone x is such that) x loves Narcissus. (from 1)

(1) is a particular instance of the existentially quantified (2), where we replace the variable with a name; and we can again infer the quantified proposition from the instance. The point of mentioning this example is to highlight that, when we generalize from a claim involving a name by using a prefixed 'some' quantifier, we don't have to replace *all* the occurrences of the name with a variable to get a valid inference using our principle.

(b) Our second principle for arguing with 'some' quantifiers, takes more explanation. Consider another obviously valid inference:

F Someone likes pizza. Whoever likes pizza likes ice cream. So someone likes ice cream.

How can we show the conclusion follows from the premisses?

Pretend for a moment that there are only two people in the domain, Jack and Jill. Then the first premiss is tantamount to a disjunction, either Jack likes pizza or Jill does. Then we can argue by cases from this disjunctive premiss. Suppose, to take the first case, Jack likes pizza and hence – by our second premiss – he likes ice cream; then someone likes ice cream. Suppose, to take the second case, Jill likes pizza, hence she likes ice cream; then again someone likes ice cream. Therefore, in either case, we can conclude someone likes ice cream. Since we are given one or other case holds, we can indeed conclude outright that someone likes ice cream.

Now drop the supposition there are just two people. Perhaps we then can't run through everyone in the domain, case by case. But we can choose a *representative* case. So take an arbitrary individual, dub them 'Alex' and suppose Alex likes pizza. Given our second premiss, it follows that Alex loves ice cream, and hence that *someone* loves ice cream. Since that conclusion *doesn't* depend on our particular choice of pizza-lover, and our first premiss tells us that there *is* at least one pizza-lover to choose, our conclusion follows outright. We could say, by analogy, that we here argue *'as if by cases'*.

Regimenting this line of reasoning, indenting when we make a temporary supposition in the now familiar way, we can set out the argument as follows:

(1) (Someone x is such that) x likes pizza. (premiss)
(2) (Everyone x is such that) if x likes pizza, x likes ice cream. (premiss)
Pick anyone in the domain, dub them 'Alex'. Then suppose
(3) Alex likes pizza. (Supposition)
(4) If Alex likes pizza, Alex likes ice cream. (from 2)
(5) Alex likes ice cream (from 3, 4)
(6) (Someone x is such that) x likes ice cream. (from 5)
But Alex was arbitrarily chosen, and our premiss (1) tells us that there is someone *who is like Alex in being, as supposed, a pizza lover. Our interim conclusion (6) doesn't depend on who Alex is. So we can argue 'as if by cases' to infer outright*
(7) (Someone x is such that) x likes ice cream. (from 1, 3–6)

What makes Alex an arbitrary representative? Again, when we rely on no special distinguishing facts about them – so we only draw on general knowledge about the domain in making inferences from the supposition (3). The inferential principle, then, is this: *Given (i) an existentially quantified proposition, and (ii) a proof that an arbitrary instance implies C (where this conclusion is independent of the particular choice of instance), then we can infer C*.

(c) Let's have a second informal example using both two principles for arguing with 'some' quantifiers. Here is another valid Aristotelian syllogism:

G Some pets are dragons. Nothing pink is a dragon. So some pets aren't pink.

Then we can deduce the conclusion from the premisses like this:

(1) (Something x is such that) x is a pet and x is a dragon. (premiss)
(2) (Everything x is such that) if x is pink, x is not a dragon. (premiss)
Pick anything in the domain, dub it 'Arb'. Then suppose
(3) Arb is a pet and Arb is a dragon. (Supposition)
(4) If Arb is pink, Arb is not a dragon. (from 2)
(5) Arb is a pet and Arb is not pink (from 3, 4)
(6) (Something x is such that) x is a pet and x is not pink. (from 5)
But Arb was arbitrarily chosen, and our premiss (1) tells us that there is something *which is like Arb in being, as supposed, a pet dragon. Our interim conclusion (6) doesn't depend on what in particular Arb is. So we can argue 'as if by cases' to infer outright*
(7) (Something x is such that) x is a pet and x is not pink. (from 1, 3–6)

(d) The four inference principles should seem intuitively natural enough, and they will be all we need for arguing with 'some' and 'every' in the guise of quantifier prefixes. So let's gather them together and restate them in a slightly tidier summary form:

31.3 Summary

Universal instantiation: from *(everything x is such that) x is F*, we can infer any instance *n is F*.

Universal generalization: from the instance *n is F*, we can infer *(everything x is such that) x is F* – so long as *n* is arbitrary enough, which means that we rely on no special facts about *n*, and the conclusion doesn't mention *n*.

Existential generalization: from the instance *n is F*, we can infer *(something x is such that) x is F*.

'As if by cases': Given *(something x is such that) x is F*, and a proof that the instance *n is F* implies the conclusion *C*, then we can infer *C* outright – so long as *n* is arbitrary enough and *C* doesn't mention *n*.

We can be content with these slightly rough versions for now. The next chapter will give formal versions, in particular clarifying the 'arbitrariness' conditions in two of the rules.

Exercises 31

Using the four quantifier principles plus propositional reasoning, give informal derivations in the style of this chapter to show that the following inferences are valid:

(1) No whales are fish. So no fish are whales.

(2) All leptons have half-integer spin. All electrons are leptons. So all electrons have half-integer spin.

(3) Some chaotic attractors are not fractals. Every Cantor set is a fractal. Hence some chaotic attractors are not Cantor sets.

(4) Some philosophers are logicians. All logicians are rational people. No rational person is a flat-farther. Therefore some philosophers are not flat-earthers.

32 Formal quantifier rules

We now turn to consider inference rules for the use of quantifiers in QL arguments. These should now look cheeringly familiar. They are just formal versions of the four intuitive semi-regimented rules we met in the last chapter.

32.1 Dummy names in QL languages

(a) We start, though, by asking: what is the expression 'Alex' – or equally, 'Arb' – doing in our informal proofs in the last chapter?

'Alex' there functions rather like the lawyer's 'John Roe' or 'Jane Doe'. It plainly isn't serving as an ordinary proper name which already has a fixed reference. On the other hand, it isn't a bound pronoun either. It is a sort of 'dummy name', or 'temporary name', or 'ambiguous name', or 'arbitrary name' – all of those labels are in use, although perhaps none is entirely happy. Another, colourless, label for the logical use is 'parameter'. We will mostly use the first and last of these labels.

When we turn to formalizing our informal quantifier proofs, then, we will want symbols to play the role of these dummy names or parameters. There is no one agreed policy for supplying such symbols. In fact there are three alternatives on the market:

(A) We can use the same symbols for free-standing parameters as we already use for variables-bound-to-quantifiers (in conventional jargon, the same symbols can appear as both 'free variables' and 'bound variables').

(B) We can use the same type of symbols for dummy names as for proper names (with just some of the names getting a fixed denotation in a given language).

(C) We can use a third, distinctive, type of symbol for dummy names.

The first policy has historically been the most common one among logicians. And it perhaps does conform best to the practice of mathematicians who casually move between using letters as dummy or temporary names ('Let m be the number of positive roots . . . ') and using letters of the same kind as quantified variables ('For any integer n, $(n+1)(n-1) = n^2 - 1$'), often leaving it to context to make it clear what the symbols are doing. However, when we go formal, overloading symbols like this can cause trouble unless we spell out careful rules for their double use.

The second policy can also be made to work with a bit of care. But once we have distinguished, at the semantic level, (i) the fixed-denotation names built into a language from (ii) its further supply of dummy names, why not highlight the distinction by using syntactically different symbols for the two styles of name?

The third policy, then, may be less economical but it does make it easier to keep track of what is going on. It conforms to the good Fregean principle of marking important differences of semantic role by using different styles of symbols. It has distinguished antecedents, e.g. in the work of the great 1930s logician Gerhard Gentzen. So (C) is the policy we adopt in this book.

(b) Given our decision, we need to augment our definition of the logical apparatus for a QL language. Thus:

> Every QL language, as well as having an unlimited supply of *variables*, starting
>
> x, y, z, . . . ,
>
> also has an unlimited supply of *dummy names*, starting
>
> a, b, c, ...
>
> Like proper names – and unlike variables – dummy names are *terms*.

In due course, we will offer a sharper definition, rather than give open-ended lists of variables and dummy names.

The *syntactic* rules for wff-building introduced in §28.2(b) and §28.6(b) can remain the same, since we have already stated them as applying to terms generally. Hence, for example, the atomic wffs of a language like QL_1 – i.e. predicates followed by the right number of terms – now include the likes of

Fa, Gb, Lbn, Ramc, . . . ,

However, expressions with variables dangling free still don't count as wffs for us. (Note, they would have done if we had instead chosen plan (A) for handling free-standing parameters, and so from the start allowed variables to be terms in §28.)

As to the *semantic* role of parameters, think of them for now as temporary names dubbing arbitrarily selected objects. We can delay giving an official story until §**??**.

(c) We will need some terminology to distinguish wffs with dummy names from those without. So, for future use, let's say:

> A QL wff without dummy names is a closed wff or a *sentence*.
>
> A wff with one or more dummy names in it will be called an open wff or an *auxiliary wff*.

Hence, for example,

$$\exists y\, Mmy,\ \exists y \forall x\, Lxy,\ \forall z(Hz \to \exists y\, Lzy),\ \forall x(Lx \to \exists y(Fy \wedge Rxyn)),\ \ldots,$$

are sample QL_1 sentences; while

$$Lmb,\ \exists y\, May,\ (Ha \to \exists y\, Lay),\ (La \to \exists y(Fy \wedge Rayn)),\ \ldots,$$

are also wffs, but auxiliary ones as they contain dummy names.

When we formalize arguments in QL languages, we will insist that these are always arguments from zero or more *sentences* as premisses to a *sentence* as final conclusion. Non-sentences with dummy names will play the same role as the informal expressions

with 'Alex' and 'Arb' in the last section, i.e. they will be used in auxiliary steps in the middle of natural deduction proofs – hence our label, auxiliary wffs.

(d) A comment. Changing between the different policy options (A), (B) and (C) for handling parameters in arguments will change which expressions count as wffs by changing the allowed non-sentences. But the class of sentences – i.e. wffs without parameters/dummy names, however they are implemented – can stay fixed across the options. And, as you would expect, *the logical relations between sentences will stay fixed too*. Hence, although we *do* have to make a policy choice here about the handling of parameters, in the most important respect it is not a very significant choice. (So don't be fazed when you find different policies being adopted in different books.)

32.2 Schematic notation, and instances of quantified offs

Before introducing our formal inference rules, it will be useful to gather together some notational and terminological conventions, old and new (compare (§28.6):

'δ' – *delta*, the Greek *d* – will now be used schematically to stand in for some dummy name.

'τ' will stand in for a term, which now can be a proper name or a dummy name.

'ξ' will, as before, stand in for some variable.

'$\alpha(\delta)$' and '$\alpha(\tau)$' will stand in for wffs which involve one or more occurrences of, respectively, the dummy name δ and the term τ.

'$\forall\xi\alpha(\xi)$' and '$\exists\xi\alpha(\xi)$' will stand in for wffs which begin with a universal or existential quantifier involving the variable ξ, and where the expression $\alpha(\xi)$ completing the wff contains one or more occurrences of the same variable ξ.

Now, our the summary versions of the four quantifier rules in the last chapter were all stated in terms of the notion of an instance of a quantified claim. So we now need a formal definition of that notion:

Suppose $\forall\xi\alpha(\xi)$ or $\exists\xi\alpha(\xi)$ is a quantified wff, τ is a term, and $\alpha(\tau)$ is the result of replacing every occurrence of ξ in $\alpha(\xi)$ by τ. Then $\alpha(\tau)$ is a wff and an *instance* of the quantified wff.

For example, here are a few quantified QL_1 wffs with some of their instances:

$\forall x\, Lxn$: Lmn, Lnn, Lan, . . . ,
$\exists z(Fz \wedge Gz)$: $(Fn \wedge Gn)$, $(Fa \wedge Ga)$, $(Fc \wedge Gc)$, . . . ,
$\forall x(Fx \to \exists y Lxy)$: $(Fm \to \exists y Lmy)$, $(Fn \to \exists y Lny)$, $(Fb \to \exists y Lby)$, ...

While the following are *not* pairs of quantified wffs and their instances:

$\neg\forall x\, Fx$: $\neg Fm$,
$(\exists z Fz \wedge Gn)$: $(Fn \wedge Gn)$,
$\forall x(Fx \to \exists y Lxy)$: $\forall x(Fx \to Lxm)$.

It is only wffs of the form $\forall\xi\alpha(\xi)$ or $\exists\xi\alpha(\xi)$ with initial quantifier prefixes which count as having instances in the defined sense.

32.3 Inference rules for '∀'

(a) We now turn to building a Fitch-style natural deduction system for proofs involving quantifiers. We start by adopting all our existing rules for dealing with the propositional connectives, now applied to wffs of our QL languages. And now we add a pair of rules for arguing with universal quantifiers.

Recall the first informal principle of inference we met in §31.1: from a universally quantified wff, we can infer *any* instance. As noted, this is often called 'universal instantiation'. But we will prefer the term ∀-*Elimination* for the formal version, in line with our previous conventions for naming rules (recall, an elimination rule for the logical operator *O* allows us to infer *from* a wff with *O* as its main operator). Using the obvious short-form label, the formal rule is:

(∀E) Given $\forall\xi\alpha(\xi)$, we can infer any instance $\alpha(\tau)$.

As before, the 'given' inputs for the application of this and other rules of inference must be *available* (in the sense of §20.6): we won't keep mentioning that.

Using a language with a suitable glossary, we can render argument **A** from the last chapter as

A′ Fm, ∀x(Fx → Gx) ∴ Gm.

And then, since we have the familiar rules for the propositional connectives available, our sketched informal argument warranting the inference simply becomes

(1)	Fm	(Prem)
(2)	∀x(Fx → Gx)	(Prem)
(3)	(Fm → Gm)	(∀E 2)
(4)	Gm	(MP 1, 3)

Here we annotate the application of the new rule (∀E) in the obvious way. Easy!

(b) The second principle we met is the 'universal generalization' rule: what applies to an arbitrary representative in a domain, applies to everything. For the formal version we will prefer the label ∀-*Introduction*. And the rule becomes this: from a wff with a dummy name – so long as this name really can be treated as temporarily dubbing an arbitrary representative because the name features in no special assumptions – then we can infer its universal generalization (where the generalization no longer mentions that representative).

So: given a wff $\alpha(\delta)$, we can universally generalize on δ – using a new variable of course! – and infer $\forall\xi\alpha(\xi)$, so long as the dummy name δ doesn't occur in any premiss or undischarged supposition. But note, if we have generalized on δ, that dummy name will no longer appear in $\forall\xi\alpha(\xi)$. Hence our formal rule is equivalent to this, which will be our official version:

(∀I) We can infer $\forall\xi\alpha(\xi)$ from a given instance $\alpha(\delta)$, so long as the dummy name δ doesn't occur in any premiss or undischarged supposition and doesn't occur in the conclusion $\forall\xi\alpha(\xi)$.

Take, then, the inference **B** from the last chapter. Rendered into a suitable ad hoc QL language, this becomes:

B′ $\forall x\,Fx,\ \forall x(Fx \to Gx) \therefore \forall x\,Gx.$

And then a formal proof warranting this inference can proceed exactly like the informal argument we gave for **B**:

(1)	$\forall x\,Fx$	(Prem)
(2)	$\forall x(Fx \to Gx)$	(Prem)
(3)	Fa	(∀E 1)
(4)	$(Fa \to Ga)$	(∀E 2)
(5)	Ga	(MP 3, 4)
(6)	$\forall x\,Gx$	(∀I 5)

We annotate the application of the new rule (∀I) in the now surely predictable way.

Similarly, we can formally render the inference **C** into a QL language like this, using an obvious glossary:

C′ $\forall x(Fx \to \neg Gx),\ \forall x(Hx \to Gx) \therefore \forall x(Fx \to \neg Hx).$

And again, we can reflect our informal derivation with a formal one (and for now, we will laboriously fill in the propositional reasoning between lines (4) and (11) below):

(1)	$\forall x(Fx \to \neg Gx)$	(Prem)
(2)	$\forall x(Hx \to Gx)$	(Prem)
(3)	$(Fa \to \neg Ga)$	(∀E 1)
(4)	$(Ha \to Ga)$	(∀E 2)
(5)	$\quad Fa$	(Supp)
(6)	$\quad \neg Ga$	(MP 5, 3)
(7)	$\quad\quad Ha$	(Supp)
(8)	$\quad\quad Ga$	(MP 7, 4)
(9)	$\quad\quad \bot$	(Abs 8, 6)
(10)	$\quad \neg Ha$	(RAA 7–9)
(11)	$(Fa \to \neg Ha)$	(CP 5–10)
(12)	$\forall x(Fx \to \neg Hx)$	(∀I 11)

Note that the dummy name 'a' does appear in the supposition at line (5), and appears again in another supposition at line (7). However – and this is crucial – by the time we get to line (11) both these suppositions have been discharged. Hence at (11) there are no active assumptions involving that dummy name. So we *are* now allowed to generalize on it using (∀I).

32.4 Inference rules for '∃'

(a) Recall the PL rules. Think of the conjuncts of a conjunction as 'instances' of the conjunction; think too of the disjuncts of a disjunction as 'instances' of the disjunction. Then the (∧E) rule tells us we can infer any instance from a conjunction; and the (∨I) rule allows us to infer a disjunction from any instance.

Now a universal quantification is like a big conjunction (see §28.8), and the (∀E) elimination rule says that we can infer any instance from a universally quantified wff. Similarly, an existential quantification is like a big disjunction. And as we would then expect, the (∃I) introduction rule will say:

> (∃I) We can infer the wff $\exists\xi\alpha(\xi)$ from any given instance $\alpha(\tau)$.

(Note: the variable ξ must be new to $\alpha(\tau)$ or $\exists\xi\alpha(\xi)$ won't be a wff.) This is our old 'existential generalization' principle from §31.2: if a named individual satisfies a certain condition, we can infer the corresponding claim that *something* satisfies that condition.

This rule than warrants the following two intuitively correct mini-proofs, which track the informal proofs for **D** and **E** from the last chapter:

(1)	(Fn ∧ Gn)	(Prem)
(2)	∃x(Fx ∧ Gx)	(∃I 1)

and

(1)	Lnn	(Prem)
(2)	∃x Lxn	(∃I 1)

(b) As before, the other existential quantifier rule – permitting us to argue 'as if by cases' – takes a bit more care to state.

Recall, the underlying idea is this. We are given that something or other is F. Pick an arbitrary representative a from the domain and suppose that a is F. If this supposition implies a conclusion C which is independent of our choice of representative a, then – since we are indeed given that something is F – we can conclude C.

So there are two conditions for an application of this rule which we need to capture. (i) We need a to be an arbitrary representative – i.e. we can't invoke any distinguishing facts about a. (ii) The conclusion C needs to be independent of our choice of representative – so C mustn't mention a.

For a formal version, we prefer the label ∃-*Elimination* (compare ∨-Elimination), and we can state the rule like this:

> (∃E) Given $\exists\xi\alpha(\xi)$, and a finished subproof from the instance $\alpha(\delta)$ as supposition to the conclusion γ – where (i) the dummy name δ is new to the proof, and (ii) γ does not contain δ – we can infer γ.

Comment: our version of condition (i) is in fact unnecessarily strong – we could e.g. allow prior occurrences of δ in earlier completed subproofs. But let's keep things simple. Adopting the rule 'always use a new dummy name when you set off on a proof by (∃E)' ensures that δ can serve to pick out an arbitrary representative.

Here then is a version of last chapter's inference **F**, once more rendered into a suitable QL language:

F′ $\exists x\, Fx,\ \forall x(Fx \to Gx) \therefore \exists x\, Gx.$

(1)	$\exists x\, Fx$	(Prem)
(2)	$\forall x(Fx \to Gx)$	(Prem)
(3)	$\quad Fa$	(Supp)
(4)	$\quad (Fa \to Ga)$	($\forall$E 2)
(5)	$\quad Ga$	(MP 3, 4)
(6)	$\quad \exists x\, Gx$	($\exists$I 5)
(7)	$\exists x\, Gx$	($\exists$E 1, 3–6)

Here we annotate the final step as you would again now expect, by giving the line number of the existentially quantified wff which we are invoking, and then noting the extent of the needed subproof from an instance of that wff with a new dummy name to the desired conclusion.

By the way, don't be too surprised that an application of the existential quantifier *elimination* rule ends up proving an existentially quantified wff at line (7)! Look again: the elimination rule in our example is being applied to the *earlier* existentially quantified wff at line (1) – it is *that* which we are arguing from, extracting information using the ($\exists$E) rule.

Similarly, we can formalize the syllogistic argument **G** like this:

G′ $\exists x(Fx \wedge Gx), \forall x(Hx \to \neg Gx) \therefore \exists x(Fx \wedge \neg Hx).$

And here is a full-dress Fitch-style proof using our rules:

(1)	$\exists x(Fx \wedge Gx)$	(Prem)
(2)	$\forall x(Hx \to \neg Gx)$	(Prem)
(3)	$\quad (Fa \wedge Ga)$	(Supp)
(4)	$\quad (Ha \to \neg Ga)$	($\forall$E 2)
(5)	$\quad Fa$	($\wedge$E 3)
(6)	$\quad Ga$	($\wedge$E 3)
(7)	$\quad\quad Ha$	(Supp)
(8)	$\quad\quad \neg Ga$	(MP 7, 4)
(9)	$\quad\quad \bot$	(5, 8)
(10)	$\quad \neg Ha$	(RAA 7–9)
(11)	$\quad (Fa \wedge \neg Ha)$	($\wedge$I 4, 10)
(12)	$\quad \exists x(Fx \wedge \neg Hx)$	($\exists$I 11)
(13)	$\exists x(Fx \wedge \neg Hx)$	($\exists$E 1, 3–12)

(c) It is, however, by now getting a bit tedious filling in routine manipulations of wffs using rules for the propositional connectives! When we can argue from some previous wffs to a new wff by using propositional reasoning, why don't we simply allow ourselves

to jump straight to that new wff? For example, here is a briefer version of the last proof – we use 'PL' on the right to signal some PL-style inference:

(1)	∃x(Fx ∧ Gx)	(Prem)
(2)	∀x(Hx → ¬Gx)	(Prem)
(3)	| (Fa ∧ Ga)	(Supp)
(4)	| (Ha → ¬Ga)	(∀E 2)
(5)	| (Fa ∧ ¬Ha)	(PL 3, 4)
(6)	| ∃x(Fx ∧ ¬Hx)	(∃I 5)
(7)	∃x(Fx ∧ ¬Hx)	(∃E 1, 3–6)

This version is not just shorter, but it highlights the quantificational part of our reasoning. So, from now on, we will occasionally cut ourselves this degree of slack by skipping over PL reasoning when writing out Fitch-style proofs for arguments using quantifiers.

(d) For another example, consider the argument:

H Only logicians are wise. Some philosophers are not logicians. All who love Aristotle are wise. Hence some people who don't love Aristotle are still philosophers.

This is valid (think about it!). Rendered into QL_1 this becomes

H′ ∀x(Hx → Gx), ∃x(Fx ∧ ¬Gx), ∀x(Lxo → Hx) ∴ ∃x(¬Lxo ∧ Fx).

So a formal derivation to warrant this inference will start

(1)	∀x(Hx → Gx)	(Prem)
(2)	∃x(Fx ∧ ¬Gx)	(Prem)
(3)	∀x(Lxo → Hx)	(Prem)

And now a general point. When premisses are a mix of universal and existential quantifications, it is a good policy to instantiate the existential(s) first. Why? Because when we use (∃E), we must always begin by instantiating the existential quantification using a dummy name *new* to the proof. So you will need to get that name into play *before* you can later use it to instantiate some relevant universal quantification.

So let's instantiate (2) as a new supposition, and then apply (∀E) to (1) and (3):

(4)	| (Fa ∧ ¬Ga)	(Supp)
(5)	| (Ha → Ga)	(∀E 1)
(6)	| (Lao → Ha)	(∀E 3)

Now, our target conclusion is '∃x(¬Lxo ∧ Fx)', which we can derive by existentially quantifying '(¬Lao ∧ Fa)'. But *that* wff follows from what we already have by propositional reasoning (how? – fill in the proof using PL rules!). Cutting ourselves some slack and skipping the details, we can conclude our proof:

(7)	| (¬Lao ∧ Fa)	(PL 4, 5, 6)
(8)	| ∃x(¬Lxo ∧ Fx)	(∃I 7)
(9)	∃x(¬Lxo ∧ Fx)	(∃E 2 4–8)

(e) For one more initial illustration of the existential rules in action, let's give a derivation for the obviously correct inference

I $(\exists x\, Fx \vee \exists x\, Gx) \therefore \exists x(Fx \vee Gx)$

The premiss is a disjunction, so we know that the overall shape of our proof is going to be an argument by cases, like this:

$(\exists x\, Fx \vee \exists x\, Gx)$	(Prem)
$\exists x\, Fx$	(Supp)
⋮	
$\exists x(Fx \vee Gx)$	
$\exists x\, Gx$	(Supp)
⋮	
$\exists x(Fx \vee Gx)$	
$\exists x(Fx \vee Gx)$	($\vee$E)

How do we fill in the dots? In each case, we have an existentially quantified supposition, and so we will expect to make use of it using the rule ($\exists$E). So we will start a new subproof by instantiating the quantified wff – using a new dummy name – and aiming for the conclusion '$\exists x(Fx \vee Gx)$'. Here's a completed proof:

(1)	$(\exists x\, Fx \vee \exists x\, Gx)$	(Prem)
(2)	$\exists x\, Fx$	(Supp)
(3)	Fa	(Supp)
(4)	$(Fa \vee Ga)$	($\vee$I 3)
(5)	$\exists x(Fx \vee Gx)$	($\exists$I 4)
(6)	$\exists x(Fx \vee Gx)$	($\exists$E 2, 3–5)
(7)	$\exists x\, Gx$	(Supp)
(8)	Gb	(Supp)
(9)	$(Fb \vee Gb)$	($\vee$I 8)
(10)	$\exists x(Fx \vee Gx)$	($\exists$I 9)
(11)	$\exists x(Fx \vee Gx)$	($\exists$E 7, 8–10)
(12)	$\exists x(Fx \vee Gx)$	($\vee$E 1, 2–6, 7–11)

32.5 Quantifier equivalences

We now have introduction and elimination rules for each of the two QL quantifiers. The good news is that *these are all the quantifier rules we need*. And if we allow ourselves to jump over merely propositional reasoning, formal quantifier proofs using these rules can often be pretty neat and natural. We will work through a further series of illustrative examples in the next chapter. For now, though, let's show that that our rules are enough to prove the following fundamental facts:

(a) The choice of variables for initial quantifiers doesn't matter (this idea was essential to our explanation of the quantifier-variable notation).
(b) The initial quantifier prefix '∀x' is equivalent to '¬∃x¬', etc., and '∃x' is equivalent to '¬∀x¬'.
(c) Two adjacent initial quantifiers of the same flavour can be interchanged, and the result is equivalent.

(a) First then, we warrant the trivial inference '∀x Fx ∴ ∀y Fy':

(1)	∀x Fx	(Prem)
(2)	Fa	(∀E 1)
(3)	∀y Fy	(∀I 2)

The reverse inference is of course proved the same way, showing that the wffs are indeed equivalent. And this pattern of proof generalizes. For any wffs $\forall\xi\alpha(\xi)$ and $\forall\zeta\alpha(\zeta)$, where the second results from the first by swapping all occurrences of the variable ξ for a variable ζ new to the expression, there will be a derivation of the shape:

(1)	$\forall\xi\alpha(\xi)$	(Prem)
(2)	$\alpha(\delta)$	(∀E 1)
(3)	$\forall\zeta\alpha(\zeta)$	(∀I 2)

Likewise, we can warrant the trivial inference '∃x Fx ∴ ∃y Fy' (and the proof similarly generalizes):

(1)	∃x Fx	(Prem)
(2)	Fa	(Supp)
(3)	∃y Fy	(∃I 2)
(4)	∃y Fy	(∃E 1, 2–3)

(b) Next, we want to warrant the inference '∀x Fx ∴ ¬∃x¬Fx'. The proof more or less writes itself. We need to derive the negation of the wff '∃x¬Fx', so we assume that wff at (2) and aim for a contradiction. And how can we now use this new existential assumption except by using (∃E)? – so we suppose an instance at (3) and continue to aim for a contradiction:

(1)	∀x Fx	(Prem)
(2)	∃x¬Fx	(Supp)
(3)	¬Fa	(Supp)
(4)	Fa	(∀E 1)
(5)	⊥	(MP 4, 3)
(6)	⊥	(∃E 2, 3–5)
(7)	¬∃x¬Fx	(RAA 2-6)

Since the absurdity sign doesn't contain 'a', the application of (∃E) conforms to the stated rule!

How do we warrant the converse inference '¬∃x¬Fx ∴ ∀x Fx'? To get the conclusion, we can try to prove 'Fa' and then universally generalize. So how can we prove 'Fa'?

Let's try that frequently useful dodge: assume the opposite and aim for a reductio!

(1)	$\neg\exists x\neg Fx$	(Prem)
(2)	$\quad\neg Fa$	(Supp)
(3)	$\quad\exists x\neg Fx$	($\exists$I 2)
(4)	$\quad\bot$	(Abs 3, 1)
(5)	$\neg\neg Fa$	(RAA 2-4)
(6)	Fa	(DN 5)
(7)	$\forall x\, Fx$	($\forall$I 6)

The final step is legal, since the dummy name 'a' in (6) occurs in no premiss or undischarged assumption.

Now for a pair of proofs showing that we can derive '$\exists x\, Fx$' and '$\neg\forall x\neg Fx$' from each other. First, we want a proof of the following shape:

$\exists x\, Fx$	(Prem)
$\quad Fa$	(Supp)
$\quad\vdots$	
$\quad\neg\forall x\neg Fx$	
$\neg\forall x\neg Fx$	($\exists$E)

Since the conclusion of the subproof begins with a negation, the obvious thing to do is assume '$\forall x\neg Fx$' and aim for a reductio. And then the proof more or less completes itself!

(1)	$\exists x\, Fx$	(Prem)
(2)	$\quad Fa$	(Supp)
(3)	$\quad\quad\forall x\neg Fx$	(Supp)
(4)	$\quad\quad\neg Fa$	($\forall$E 3)
(5)	$\quad\quad\bot$	(Abs 2, 4)
(6)	$\quad\neg\forall x\neg Fx$	(RAA 3–5
(7)	$\neg\forall x\neg Fx$	($\exists$E 1, 2–6)

The derivation in the reverse direction is rather more interesting. Our premiss is '$\neg\forall x\neg Fx$'; how are we going to derive the desired conclusion '$\exists x\, Fx$'? Again, what else can we do but assume the opposite and aim for a reductio? Which will give us a proof of this skeletal shape:

$\neg\forall x\neg Fx$	(Prem)
$\quad\neg\exists xFx$	(Supp)
$\quad\vdots$	
$\quad\bot$	
$\neg\neg\exists xFx$	(RAA)
$\exists x\, Fx$	(DN)

To fill in the dots, we will need to make another temporary assumption. Which one? Well, to get a contradiction we want to prove '∀x¬Fx' which we will get by generalizing from something like '¬Fa'. So let's suppose 'Fa' and aim for a contradiction:

(1)	¬∀x¬Fx	(Prem)
(2)	¬∃xFx	(Supp)
(3)	Fa	(Supp)
(4)	∃x Fx	(∃I 3)
(5)	⊥	(Abs 4, 2)
(6)	¬Fa	(RAA 3–5)
(7)	∀x¬Fx	(∀I 6)
(8)	⊥	(Abs 7, 1)
(9)	¬¬∃xFx	(RAA 2-8)
(10)	∃x Fx	(DN 9)

(c) Finally, here are a couple of very simple proofs illustrating how we can swap the order of two initial quantifiers of the same flavour. The universal case is easy:

(1)	∀x∀y Lxy	(Prem)
(2)	∀y Lay	(∀E 1)
(3)	Lab	(∀E 2)
(4)	∀x Lxb	(∀I 3)
(5)	∀y∀x Lxy	(∀I 4)

Next, we want to warrant the inference '∃x∃y Lxy ∴ ∃y∃x Lxy'. Given the existential premiss, we need to instantiate it in a supposition to prepare for an argument by (∃E). But this instance is another existentially quantified wff, so we need next to take an instance of *that*. Then the proof to our desired conclusion more or less completes itself:

(1)	∃x∃y Lxy	(Prem)
(2)	∃y Lay	(Supp)
(3)	Lab	(Supp)
(4)	∃x Lxb	(∃I 3)
(5)	∃y∃x Lxy	(∃I 4)
(6)	∃y∃x Lxy	(∃E 3–5)
(7)	∃y∃x Lxy	(∃E 1, 2–6)

Do pause, however, to double-check that the applications of (∃E) here indeed obey the official rules!

(d) The results in this section can all be generalized. So we could augment our QL proof system with *derived rules* in the sense of §23.2. For example, we could add a rule allowing ourselves to change variables as in (a). Or a rule allowing ourselves to swap initial quantifiers of the same flavour as in (c). But probably the most useful short-cut rules to add would be this pair:

(∀∃) Given one of $\forall\xi\alpha(\xi)$ and $\neg\exists\xi\neg\alpha(\xi)$, you can infer the other.

(∃∀) Given one of $\exists\xi\alpha(\xi)$ and $\neg\forall\xi\neg\alpha(\xi)$, you can infer the other.

However, we won't officially adopt these additional rules (though feel free to use them if you want). For our principal aim is to develop the *understanding* of the core principles of our Fitch-style proof system. Our focus now is on understanding the quantificational part of the system, and that's why we are happy now occasionally to skip over the details of PL reasoning in the middle of proofs so as not to obscure a proof's quantificational core. But, as before, we are not particularly concerned to construct ever more complicated proofs for which derived quantification rules might come in handy as short-cuts.

32.6 QL theorems

(a) Finally in this chapter, let's pause to note that our new rules will generate new *theorems*, i.e. new wffs that can be proved from *no* premisses.

Of course, whenever we have a proof from the premiss α to a conclusion γ, there will be a corresponding theorem of the form $(\alpha \to \gamma)$ – compare §24.5. So, for example, the following is a theorem:

J $((\exists x\, Fx \vee \exists x\, Gx) \to \exists x(Fx \vee Gx))$.

Just suppose the antecedent, derive the consequent as in our proof for **I**, and apply (CP).

There are, however, more interesting cases of QL theorems where the main logical operator is not a conditional. For example, consider

K $\forall x((Fx \wedge Hx) \to Fx)$,

which in QL_1 expresses the logical truth that whoever is a wise philosopher is indeed a philosopher. So this certainly *ought* to be demonstrable just from the rules governing the logical operators. And it is:

(1)	|	
(2)	| (Fa ∧ Ha)	(Supp)
(3)	| Fa	(∀E 2)
(4)	((Fa ∧ Ha) → Fa)	(CP 2–3)
(5)	∀x((Fx ∧ Hx) → Fx)	(∀I 3)

Note that this is a correctly formed proof. The temporary supposition involving the dummy name is discharged before we get to universally generalize on that name.

(b) Here is a much more interesting example to finish with. The following is a theorem (in any language containing a binary predicate 'L'):

$$\neg\exists x\forall y(Lxy \leftrightarrow \neg Lyy).$$

Or, rather, unpacking our double-arrow shorthand for the biconditional, *this* is a theorem:

L $\neg\exists x\forall y((Lxy \to \neg Lyy) \wedge (\neg Lyy \to Lxx))$.

If we assume this wff without its initial negation sign, then we can very quickly derive absurdity:

(1)	|	
(2)	| ∃x∀y((Lxy → ¬Lyy) ∧ (¬Lyy → Lxy))	(Prem)
(3)	| | ∀y((Lry → ¬Lyy) ∧ (¬Lyy → Lry))	(Supp)
(4)	| | ((Lrr → ¬Lrr) ∧ (¬Lrr → Lrr))	(∀E 2)
(5)	| | ⊥	(PL 3)
(6)	| ⊥	(∃E 2, 3–5)
(7)	∃x∀y((Lxy → ¬Lyy) ∧ (¬Lyy → Lxy))	(RAA 2–6)

For note, (i) we can instantiate the quantification (3) with *any* term, including one already in the wff, to get (4). (ii) (4) tells us that 'Lrr' is true if and only it is false, which is impossible; unsurprisingly, there is a simple PL proof of absurdity from (4). (iii) Again, (∃E) is correctly applied to derive (6) – '⊥' doesn't contain 'r'!

Now suppose that we (re)interpret 'L' as ① *shaves* ②, and take the domain to be *the men in some particular village*. Then **J** tells us that there is no man in the village who shaves all and only those who do not shave themselves. Sometimes this is called the Barber Paradox – but there is no paradox here, just a logical theorem that there can be no such person!

Suppose that we instead interpret 'L' as ① *is a member of* ②, and take the domain to be the universe of *sets*. Then **J** tells us that there is no set which has as its members just those sets which are not members of themselves. Think of a set which doesn't contain itself as a *normal* set: then we have shown that there is no set of all normal sets. This is, famously, Russell's Paradox. And this time the label 'paradox' is perhaps more appropriate.

For '① is a normal set' seems a perfectly sensible unary predicate. And it is a rather plausible principle that, given a sensible unary predicate, we can gather the things that satisfy the predicate into a set (the predicate will have this set as its extension). So it is a surprise to find, purely as a matter of logic, there can be no set of normal sets – our plausible principle can't be applied across the board.

32.7 Summary

To formalize quantifier arguments, we need symbols in our QL languages available for use as dummy names. We take the option of using special-purpose symbols, lower-case letters from the beginning of the alphabet. Syntactically, these dummy names behave the same way as proper names, providing more terms.

Guided by the informal rules summarized at the end of the previous chapter, we presented pairs of introduction and elimination rules for the formal universal and existential quantifiers. (We will not restate the formal rules now, but instead we will set them out diagrammatically at the beginning of the next chapter.)

Using these rules, we have found proofs for some simple syllogistic arguments, and also demonstrated some basic quantifier equivalences – we can swap variables, swap the order of quantifiers of the same kind, and swap $\forall\xi$ for $\neg\exists\xi\neg$, etc.

Exercises 32

33 More QL proofs

We continue to explore the Fitch-style natural deduction system for arguing in QL languages that results from adding our four quantifier rules to the familiar rules for the propositional connectives. We start by stressing again the need for the restrictions on the dummy names used in applications of the rules (∀I) and (∃E). Then we work through some more examples of quantifier proofs, commenting on various issues as we go.

33.1 The QL rules again – and how to misuse them

(a) Let's gather together the introduction and elimination rules for the two quantifiers, now in diagrammatic form, arranged to bring out some relations between the rules:

Rules for quantifiers

(∀E)
$\forall\xi\alpha(\xi)$
$\vdots$
$\alpha(\tau)$

(∃I)
$\alpha(\tau)$
$\vdots$
$\exists\xi\alpha(\xi)$

(∀I)
$\alpha(\delta)$
$\vdots$
$\forall\xi\alpha(\xi)$

(∃E)
$\exists\xi\alpha(\xi)$
$\vdots$
$\alpha(\delta)$ | $\vdots$ γ
γ

In all these rules, $\alpha(\tau)$ or $\alpha(\delta)$ is an instance of the corresponding quantified wff; as usual τ can be any kind of term, but δ must be a dummy name.

The following restrictions must be observed on the dummy names δ:

For (∀I), δ must not appear in any premiss or undischarged supposition or in the conclusion $\forall\xi\alpha(\xi)$.
For (∃E), δ must be new to the proof and must not appear in the conclusion γ.

(b) The first two rules are entirely straightforward. The second two are a bit trickier to handle, being governed by the crucial restrictions on the use of dummy names which we

motivated in the last chapter. The restrictions are in fact reasonably easy to remember if you keep those motivations clearly in mind. But it will help to fix ideas if in the rest of this section we now consider some ways things can go disastrously wrong if we offend against those restrictions.

So first, let's give a fake derivation from '$\exists x\, Fx$' to '$\forall x\, Fx$':

(1)	$\exists x\, Fx$	(Prem)
(2)	$\quad Fa$	(Supp)
(3)	$\quad \forall x\, Fx$	(!!$\forall$I 2)
(4)	$\forall x\, Fx$	($\exists$E 1, 2–3)

Intuitively, things go wrong at (3): from e.g. the supposition that some chosen individual is a logician, we can't infer that everyone is! Formally, the hopeless move is blocked by the restriction that says we can't universally generalize on a dummy name if it features in an undischarged supposition.

Here's another 'proof' for the same bad inference:

(1)	$\exists x\, Fx$	(Prem)
(2)	$\quad Fa$	(Supp)
(3)	$\quad Fa$	(Iter 2)
(4)	Fa	(!!$\exists$E 1, 2–3)
(5)	$\forall x\, Fx$	($\forall$I 4)

This time the gruesome mistake is at line (4). Our subproof doesn't reach a conclusion independent of our choice of instance for (1), so we can't argue 'as if by cases'. Formally, what's gone wrong is that we have tried to apply ($\exists$E) when the subproof ends with a wff which still contains the relevant dummy name introduced at the beginning of the subproof.

Now let's 'prove' that, if everyone loves someone, then someone loves themself – so we will argue from '$\forall x \exists y\, Lxy$' to '$\exists x\, Lxx$':

(1)	$\forall x \exists y\, Lxy$	(Prem)
(2)	$\exists y\, Lay$	($\forall$E 1)
(3)	$\quad Laa$	(Supp)
(4)	$\quad \exists x\, Lxx$	($\exists$I 3)
(5)	$\exists x\, Lxx$	(!!$\exists$E 2 3–4)

What's gone wrong here? At line (2), we have in effect picked an individual *a* from the domain and we know from (1) that *a* loves *someone* or other. We can now go on chose a representative beloved for *a* at (3) – but we are not entitle to continue the argument by taking that representative to be *a* again!

Here's a similar case. We will 'prove' that, given first that someone is a logician and second that someone is irrational, it follows that there is an irrational logician.

Using the obvious glossary, here then is a fake derivation from '$\exists x\, Fx$' and '$\exists x\, Gx$' to '$\exists x(Fx \wedge Gx)$':

(1)	∃x Fx	(Prem)
(2)	∃x Gx	(Prem)
(3)	Fa	(Supp)
(4)	Ga	(Supp)
(5)	(Fa ∧ Ga)	(∧I, 3, 4)
(6)	∃x(Fx ∧ Gx)	(∃I, 5)
(7)	∃x(Fx ∧ Gx)	(!!∃E 2, 4–6)
(8)	∃x(Fx ∧ Gx)	(∃E 1, 3–7)

What's gone wrong this time? We've picked a representative logician *a* at line (3). Then we've picked a representative irrational person at line (4) – but we can't assume that this is *a* again.

Formally, in both these last two fake proofs, we have broken the rule which tells us that when we instantiate an existential quantification to start a subproof for (∃E) we must always use a dummy name new to the proof.

(c) Let's consider another pair of inferences:

∃y∀xLxy ∴ ∀x∃yLxy,
∀x∃yLxy ∴ ∃y∀xLxy.

The first is valid: if there is a particular someone whom everyone loves, then certainly everyone loves someone or other. But the reverse second inference is invalid.

Here then is a genuine proof for the first inference:

(1)	∃y∀x Lxy	(Prem)
(2)	∀xLxa	(Supp)
(3)	Lba	(∀E 2)
(4)	∃y Lby	(∃I 3)
(5)	∀x∃y Lxy	(∀I 4)
(6)	∀x∃y Lxy	(∃E 1, 2–5)

Check that this proof is correctly formed.

And now here is a 'proof' of the reverse inference:

(1)	∀x∃y Lxy	(Prem)
(2)	∃yLay	(∀E 1)
(3)	Lab	(Supp)
(4)	∀x Lxb	(!!∀I 3)
(5)	∃y∀x Lxy	(∃I 4)
(6)	∃y∀x Lxy	(∃E 1, 2–5)

Here things go disastrously wrong at line (4). Even though 'a' was introduced as picking out an arbitrary individual, we can't now universally generalize on it. Why?

(d) Let's have one last fake proof. We will 'demonstrate' that if everyone loves themself then everyone loves everyone – i.e. we will try to argue from '∀x Lxx' to '∀x∀y Lxy':

(1)	∀x Lxx	(Prem)
(2)	Laa	(∀E 1)
(3)	∀y Lay	(!!∀I 2)
(4)	∀x∀y Lxy	(∀I 3)

We go wrong at (3). From the assumption that some arbitrarily selected individual *a* loves themself, we can't infer that *a* loves everyone. Formally, we have offended against the rule that when we universally generalize on a dummy name, we have to replace *all* the occurrences of the name with the relevant variable. Equivalently, when we derive a universal quantification from an instance with a dummy name, that name must not appear in the quantified wff.

The moral of all our examples? Obeying those restrictions on the use of dummy names in the rules (∀I) and (∃E) is *essential* if you are to construct valid proofs.

33.2 Old and new logic: three proofs

In this section we look at three kinds of arguments which 'traditional' logicians had trouble with, to show that such arguments can be dealt with easily by our QL proof system which incorporates Fregean insights.

(a) Take first the simple two-step argument

A Everyone loves themself. So Maldwyn loves Maldwyn. So someone loves Maldwyn.

Why does this cause trouble for a logic of generality rooted in the Aristotelian tradition? Consider the *second* inference. To explain its validity, the traditional logician – relying on a subject/predicate analysis of propositions – needs to discern a common (unary) predicate in its two propositions, namely 'loves Maldwyn'. But then, if 'Maldwyn loves Maldwyn' involves just *that* predicate, exactly why does it follow from the premiss of the *first* inference – which involves the quite different predicate 'loves themself'?

Formalized into a suitable QL language, however, the two-step inference **A** becomes this:

(1)	∀x Lxx	(Prem)
(2)	Lmm	(∀E 1)
(3)	∃x Lxm	(∃I 2)

The predicate-first notation is a mere matter of style. What is crucial is, first, that the middle proposition is now analysed at (2) as involving a *binary* predicate 'L' with two slots to be filled. And second, Frege's great insight, we need to understand a quantifier as something that can be tied to different numbers of places in the same predicate. Then we can see how the very same predicate can feature in both (1) and (3).

(b) Consider next

B Either every wombat is a mammal or every wombat is a marsupial. Hence every wombat is either a mammal or a marsupial.

The main logical operator of the premiss is the disjunction. The main operator of the conclusion is the quantifier. To deal with this inference, we need a logic than can simultaneously handle sentential connectives and quantifiers.

Ancient Stoic logic copes with some propositional reasoning; Aristotelian logic copes with some quantifier reasoning. Much later – to continue the cartoon history – George Boole in the early nineteenth century came up with a logical algebra which could be interpreted two ways. It could be read as being about 'and', 'or' and 'not' (that's why we still speak of Boolean connectives) or alternatively as being about Aristotelian syllogisms. But Boole's system still couldn't deal with both at the same time.

By contrast, our Frege-based treatment of the quantifiers combines with a logic for the connectives to allow us to argue by cases like this (using the obvious glossary):

(1)	$(\forall x(Fx \to Gx) \vee \forall x(Fx \to Hx))$	(Prem)
(2)	$\quad \forall x(Fx \to Gx)$	(Supp)
(3)	$\quad (Fa \to Ga)$	($\forall$E 2)
(4)	$\quad (Fa \to (Ga \vee Ha))$	(PL 3)
(5)	$\quad \forall x(Fx \to (Gx \vee Hx))$	($\forall$I 4)
(6)	$\quad \forall x(Fx \to Hx)$	(Supp)
(7)	$\quad (Fa \to Ha)$	($\forall$E 6)
(8)	$\quad (Fa \to (Ga \vee Ha))$	(PL 7)
(9)	$\quad \forall x(Fx \to (Gx \vee Hx))$	($\forall$I 8)
(10)	$\forall x(Fx \to (Gx \vee Hx))$	($\exists$E 1, 2–5, 6–9)

For brevity, we allow ourselves to skip over routine PL reasoning at lines (4) and (8). (See argument **I** in the last chapter for a similar example.)

(c) The most intractable problem for traditional logic, however, was dealing with arguments involving propositions with quantifiers embedded inside the scope of other quantifiers (as we would now put it). Consider for example the following inference

C Every horse is a mammal. Hence every horse's tail is a mammal's tail.

To bring out the multiple generality in the conclusion, read it as 'Everything which is the tail of some horse is the tail of some mammal'. The inference is then evidently valid. Medieval logicians in the Aristotelian tradition, for all their insightful ingenuity, struggled to cope. But there's a straightforward QL proof.

So, let's adopt a QL language with the following glossary:

F: ① is a horse,
G: ① is a mammal,
T: ① is a tail belonging to ②;

and take the domain to be inclusive enough, e.g. all terrestial physical objects. Then we can render **C** as

C′ $\forall x(Fx \to Gx) \therefore \forall x(\exists y(Txy \wedge Fy) \to \exists y(Txy \wedge Gy))$.

Now, the conclusion here is universally quantified, so we will expect it to be derived by universal quantifier introduction from a wff involving some dummy name, like this:

$\forall x(Fx \to Gx)$	(Prem)
⋮	
$(\exists y(Tay \wedge Fy) \to \exists y(Tay \wedge Gy))$	
$\forall x(\exists y(Txy \wedge Fy) \to \exists y(Txy \wedge Gy))$	($\forall$I)

The penultimate line is a conditional; so we will presumably want to assume its antecedent and then derive its consequent. And this antecedent is an existentially quantified wff; so the natural thing to do is to assume an instance of it, with a view to arguing by an application of ($\exists$E). Which will make the overall shape of the proof like this:

$\forall x(Fx \to Gx)$	(Prem)
$\quad \exists y(Tay \wedge Fy)$	(Supp)
$\quad\quad (Tab \wedge Fb)$	(Supp)
$\quad\quad$ ⋮	
$\quad\quad \exists y(Tay \wedge Gy)$	
$\quad \exists y(Tay \wedge Gy)$	($\exists$E)
$(\exists y(Tay \wedge Fy) \to \exists y(Tay \wedge Gy))$	(CP)
$\forall x(\exists y(Txy \wedge Fy) \to \exists y(Txy \wedge Gy))$	($\forall$I)

Do pause at this stage to double check how the applications of the two rules ($\exists$E) and ($\forall$I) obey the restrictions on their relevant dummy names 'a' and 'b'.

So given that outline proof, we now simply need to join up the remaining dots:

(1)	$\forall x(Fx \to Gx)$	(Prem)
(2)	$\quad \exists y(Tay \wedge Fy)$	(Supp)
(3)	$\quad\quad (Tab \wedge Fb)$	(Supp)
(4)	$\quad\quad (Fb \to Gb)$	($\forall$E 1)
(5)	$\quad\quad (Tab \wedge Gb)$	(PL 3, 4)
(6)	$\quad\quad \exists y(Tay \wedge Gy)$	($\exists$I 5)
(7)	$\quad \exists y(Tay \wedge Gy)$	($\exists$E 2 3–6)
(8)	$(\exists y(Tay \wedge Fy) \to \exists y(Tay \wedge Gy))$	(CP 2–7)
(9)	$\forall x(\exists y(Txy \wedge Fy) \to \exists y(Txy \wedge Gy))$	($\forall$I 8)

(d) A comment on our handling of the last example. In regimenting the informal argument **C**, we chose to use a language including the slightly odd predicate

T: ① is a tail belonging to ②.

You might perhaps have expected to find instead the following two predicates:

H: ① is a tail,
B: ① belongs to ②.

And then instead of 'Txy', you would expect to use '(Hx ∧ Bxy)', etc.

Well, yes, it could be said that our predicate 'T' is unnatural. And if we also wanted to regiment other arguments about animals and their parts, it might well be that we'll

need to have predicates like 'H' and 'B' separately available. However, in the present context, replacing occurrences of 'Txy' by '(Hx ∧ Bxy)' etc. just produces clutter. Our proof written out this way would make *no* use of the internal structure of this complex expression, so why introduce it?

When we construct an ad hoc language for regimenting some argument(s), it is on the whole good policy to dig down only just as far as we need. As Quine famously puts it,

> "A maxim of shallow analysis prevails: expose no more logical structure than seems useful for the deduction or other inquiry at hand."

33.3 Five more QL proofs

(a) In §4.5, we met the following inference, and informally showed that it is valid.

D No girl loves any unreconstructed sexist; Caroline is a girl who loves whoever loves her; Henry loves Caroline; hence Henry is not an unreconstructed sexist.

Using a QL language with a suitable glossary, and rendering the 'no' premiss with a universal quantifier, we can translate the inference like this:

D′ $\forall x(Gx \to \forall y(Fy \to \neg Lxy)),\ (Gm \wedge \forall x(Lxm \to Lmx)),\ Lnm \ \therefore\ \neg Fn$

To construct a formal derivation of the conclusion, the obvious thing to do is to assume 'Fn' and aim for a contradiction. So think through the following proof:

(1)	$\forall x(Gx \to \forall y(Fy \to \neg Lxy))$	(Prem)
(2)	$(Gm \wedge \forall x(Lxm \to Lmx))$	(Prem)
(3)	Lnm	(Prem)
(4)	$\quad Fn$	(Supp)
(5)	$\quad Gm$	(∧E 2)
(6)	$\quad \forall x(Lxm \to Lmx)$	(∧E 2)
(7)	$\quad (Gm \to \forall y(Fy \to \neg Lmy))$	(∀I 1)
(8)	$\quad \forall y(Fy \to \neg Lmy)$	(MP 5, 7)
(9)	$\quad (Fn \to \neg Lmn)$	(∀E 8)
(10)	$\quad \neg Lmn$	(MP 4, 9)
(11)	$\quad (Lnm \to Lmn)$	(∀E 6)
(12)	$\quad Lmn$	(MP 3, 11)
(13)	$\quad \bot$	(Abs 12, 10)
(14)	$\neg Fn$	(RAA 4–13)

What have we done here after line (4)?

At line (2) there is a conjunction, and we'll evidently want to use its conjuncts – let's disassemble it straight away, giving us lines (5) and (6).

We have a universal quantification at line (1), and we can instantiate it with any term in our formal language. But we will in fact want to instantiate it with the name 'm' to get the conditional with antecedent 'Gm', since we *already* have 'Gm' at line (5). So this motivates inferring the instance (7), and then we can apply modus ponens.

We can likewise instantiate the new universal quantification at (8) in a way that similarly links up to 'Fn' earlier in the proof, to get (9) and hence (10).

And *now* we are almost done. (3) and (10) don't by themselves contradict each other. However, it is easy to see that we can instantiate (6) in the right way to get (11) and hence absurdity. Not for the first time, then, a proof almost writes itself!

(b) For our second example, we return to another silly inference that we have met before:

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

We read the first premiss as saying: take anyone x, then if x is a lover (i.e. loves someone), then everyone loves x. So, adopting a language with the obvious glossary, we can regiment this argument as

E′ $\forall x(\exists y\, Lxy \rightarrow \forall y\, Lyx),\ Lrj \therefore \forall x\, Lxj$

And now we can formally replicate that informal multi-step argument we gave back in §4.1:

(1)	$\forall x(\exists y\, Lxy \rightarrow \forall y\, Lyx)$	(Prem)
(2)	Lrj	(Prem)
(3)	$(\exists y\, Lry \rightarrow \forall y\, Lyr)$	(∀E 1)
(4)	$\exists y\, Lry$	(∃I 2)
(5)	$\forall y\, Lyr$	(MP 4, 3)
(6)	Ljr	(∀E 5)
(7)	$(\exists y\, Ljy \rightarrow \forall y\, Lyj)$	(∀E 1)
(8)	$\exists y\, Ljy$	(∃I 6)
(9)	$\forall y\, Lyj$	(MP 8, 7)

Again, at each step in this proof so far, we just do the natural thing. At step (3) we will want to instantiate the quantified premiss, and the obvious option is to use the name 'r' to give us a conditional with an antecedent that connects with (2). That quickly takes us to (5), and what is the natural term to instantiate *that* with? Surely 'j' this time. And so the proof continues . . . .

We haven't quite arrived at our target conclusion, however, as we regimented the conclusion of **E** using 'x', the first available variable for the quantification. Bother! We could retranslate that conclusion as '$\forall y\, Lyj$': but that's a bit sneaky. So let's keep ourselves honest and finish up by using the variable-changing trick we met in §32.5:

(10)	Laj	(∀E 9)
(11)	$\forall x\, Lxj$	(∀I 10)

(c) For our next example, consider the following inference (the topic, let's suppose, is the logic class and the questions on a particular test):

F If everyone in the class can answer *every* question, then some questions are too easy. So at least one person in the class is such that, if even *they* can answer every question, then some questions are too easy.

That is valid. Let's show this by a QL proof.

We might be tempted to reflect all the surface structure of the premiss and conclusion, and come up with a QL translation along the following lines:

$$(\forall x(Cx \to \forall y(Qy \to Axy)) \to \exists x(Qx \wedge Ex)) \therefore \exists x((Cx \wedge \forall y(Qy \to Axy)) \to \exists x(Qx \wedge Ex))$$

Here the quantifier runs over some inclusive domain containing both people and questions, and we are using a QL language with a glossary like

C: ① is in the class,
Q: ① is a question,
E: ① is too easy,
A: ① can answer ②.

But this depth of analysis is unnecessary. (i) The internal structures of '$\forall y(Qy \to Axy)$' and of '$\exists x(Qx \wedge Ex)$' do no work in the argument. And (ii) since the other quantifiers are restricted to people in the class, we might as well take those people to make up the whole universe of discourse. So, following Quine's maxim of shallow analysis, let's take a simpler language with the following glossary:

F: ① can answer every question,
P: Some of the questions (in the test) are too easy.
Domain: people in the class

Remember, we allow QL languages to have propositional letters (predicates of arity zero – see §**??**(d)). And now our translated inference looks *very* much more manageable!

F′ $(\forall x\, Fx \to P) \therefore \exists x(Fx \to P)$

We've seen inferences of this form in §29.6. And here's one corresponding formal derivation (why is each step the natural one to make?)

(1)	$(\forall x\, Fx \to P)$	(Prem)
(2)	$\neg\exists x(Fx \to P)$	(Supp)
(3)	$(Fa \to P)$	(Supp)
(4)	$\exists x(Fx \to P)$	($\exists$I 3)
(5)	$\bot$	(Abs 4, 2)
(6)	$\neg(Fa \to P)$	(RAA 3–5)
(7)	Fa	(PL 6)
(8)	$\neg P$	(PL 6)
(9)	$\forall x\, Fx$	($\forall$I 7)
(10)	P	(MP 69, 1)
(11)	$\bot$	(Abs 10, 8)
(12)	$\neg\neg\exists x(Fx \to P)$	(RAA 2–11)
(13)	$\exists x(Fx \to P)$	(DN 12)

(d) Next, here is an argument we met in the very first chapter, and then informally argued to be valid in §3.1:

G Some philosophy students admire all logicians. No philosophy student admires anyone irrational. So no logician is irrational.

We can translate the 'no' propositions using universal or existential quantifiers. Let's start by going the first way. Choosing formal predicate letters to match the English, we can then render the argument like this:

G′ $\exists x(Px \wedge \forall y(Ly \rightarrow Axy)),\ \forall x(Px \rightarrow \forall y(Iy \rightarrow \neg Axy)) \ \therefore\ \forall x(Lx \rightarrow \neg Ix)$

(By the way, don't take the translations on trust: do check them!)

Using the rule of thumb 'instantiate existential premisses first', the overall shape of the proof we are looking for is:

$\exists x(Px \wedge \forall y(Ly \rightarrow Axy))$	(Prem)
$\forall x(Px \rightarrow \forall y(Iy \rightarrow \neg Axy))$	(Prem)
$(Pa \wedge \forall y(Ly \rightarrow Aay))$	(Supp)
⋮	
$\forall x(Lx \rightarrow \neg Ix)$	
$\forall x(Lx \rightarrow \neg Ix)$	(∃E)

Unpack the conjuncts in the supposition. And then it is natural to instantiate the second premiss using the dummy name 'a'. Which gets us painlessly to line (7) here:

(1)	$\exists x(Px \wedge \forall y(Ly \rightarrow Axy))$	(Prem)
(2)	$\forall x(Px \rightarrow \forall y(Iy \rightarrow \neg Axy))$	(Prem)
(3)	$(Pa \wedge \forall y(Ly \rightarrow Aay))$	(Supp)
(4)	Pa	(∧E 3)
(5)	$\forall y(Ly \rightarrow Aay)$	(∧E 3)
(6)	$(Pa \rightarrow \forall y(Iy \wedge \neg Aay))$	(∀E 2)
(7)	$\forall y(Iy \rightarrow \neg Aay)$	(MP 4, 6)

So the remaining task is to get from (5) and (7) to the target conclusion of the subproof, i.e. to derive '$\forall x(Lx \rightarrow \neg Ix)$'.

But this is relatively simple (compare **C′** in the previous chapter). We instantiate the two universal quantifiers (not with 'a' again because we want an arbitrary instance we can generalize on later). And then continue:

(8)	$(Lb \rightarrow Aab)$	(∀E 5)
(9)	$(Ib \rightarrow \neg Aay)$	(∀E 7)
(10)	$(Lb \rightarrow \neg Ib)$	(PL 8, 9)
(11)	$\forall x(Lx \rightarrow \neg Ix)$	(∀I 10)
(12)	$\forall x(Lx \rightarrow \neg Ix)$	(∃E 1, 3–11)

(e) For a last example, it is perhaps worth considering how things go if we alternatively translate the 'no' propositions in **G** using negated existential quantifiers, as in

G″ $\exists x(Px \wedge \forall y(Ly \rightarrow Axy)),\ \neg\exists x\exists y(Px \wedge (Ir \wedge Axy)) \ \therefore\ \neg\exists x(Lx \wedge Ix)$

We will talk through how to discover a proof.

But before proceeding, let's remember not to lose sight of the point we first stressed in §20.5. The key thing, always, is to make sure you *understand* the general principles deployed in proofs. For us, as philosophers, proof-discovery remains secondary.

In this case, given **G″**'s negative conclusion, the obvious strategy is to assume '$\exists x(Lx \wedge Ix)$' at line (3) and aim for absurdity, so we can use a reductio argument.

We will then have *two* existential assumptions in play, at lines (1) and (3): so instantiate them in turn at lines (4) and (5). We get two conjunctions as a result. So next disassemble those. And we reach the following stage:

(1)	$\exists x(Px \wedge \forall y(Ly \rightarrow Axy))$	(Prem)
(2)	$\neg\exists x\exists y(Px \wedge (Iy \wedge Axy))$	(Prem)
(3)	$\exists x(Lx \wedge Ix)$	(Supp)
(4)	$(Pb \wedge \forall y(Ly \rightarrow Aby))$	(Supp)
(5)	$(La \wedge Ia)$	(Supp)
(6)	Pb	($\wedge$E 4)
(7)	$\forall y(Ly \rightarrow Aby)$	($\wedge$E 4)
(8)	La	($\wedge$E 5)
(9)	Ia	($\wedge$E 5)

What next? How do we derive '$\bot$'? Well, obviously we will need to use the quantified wff at (7). What shall we instantiate it with? The natural choice is 'a' as that will give as a conditional whose antecedent we will already have at line (8). So we can continue:

(10)	$(La \rightarrow Aba)$	($\forall$E 7)
(11)	Aba	(MP 8, 10)

And now the end is in sight. For we can now conjoin (11) with some earlier lines to form the conjunction

(12)	$(Pb \wedge (Ia \wedge Aba))$	(PL 6, 9, 11)

But this is flatly inconsistent with (2). Bring out the contradiction, and the derivation completes itself, with the Fitch-style layout beautifully keeping track of the ins and outs of the proof:

(13)	$\exists y(Pb \wedge (Iy \wedge Aby))$	($\exists$I 12)
(14)	$\exists x\exists y(Px \wedge (Iy \wedge Axy))$	($\exists$I 13)
(15)	$\bot$	(Abs 14, 2)
(16)	$\bot$	($\exists$E 3, 5–15)
(17)	$\bot$	($\exists$E 1, 4–16)
(18)	$\neg\exists x(Lx \wedge Ix)$	(RAA 3–17)

Which wasn't too hard! Just check that the two applications of ($\exists$E) are correctly done.

33.4 Summary

We have restated the formal introduction and elimination quantifier rules in a diagrammatic form. Note in particular the (natural and intuitively motivated) restrictions on the use of dummy names in the rules (∀I) and (∃E). Obey these restrictions on pain of constructing fallacious fake proofs!

Pre-Fregean logic had serious difficulties coping with arguments that involve both connectives and quantifiers, or which involve propositions with multiple quantifiers. We saw how modern quantifications logic can cope smoothly and naturally with such arguments.

The key bit of practical advice in proof-construction: instantiate existential quantifiers – to start subproofs for use in applications of (∃E) – before instantiating relevant universal quantifiers.

34 Empty domains?

We stipulated that the domain of quantification for a QL language must contain at least one object. This might seem to be a very modest requirement – after all, we surely don't want to be talking about nothing at all. But still, we need to comment.

34.1 On the use of dummy names

In §31.1 we considered this obviously valid inference

A Everyone likes pizza. Whoever likes pizza likes ice cream. So everyone likes ice cream.

How might we derive the conclusion from the premisses with an informal proof? We gave an argument starting like this:

(1) (Everyone x is such that) x likes pizza. (premiss)
(2) (Everyone x is such that) if x likes pizza, then x likes ice cream. (premiss)
Now pick any person in the domain, temporarily dub them 'Alex'. Then:
(3) Alex likes pizza (from 1)
(4) . . .

But hold on! What if the domain is empty? Then there is no one in the domain to pick out and dub with a temporary name at line (3). So our informal derivation at this point in effect presupposes that the domain is non-empty.

In §32.3, we considered a formal version of the same inference

A′ $\forall x\,Fx,\ \forall x(Fx \to Gx) \therefore\ \forall x\,Gx.$

And we gave a formal proof warranting this inference. It starts:

(1′)	$\forall x\,Fx$	(Prem)
(2′)	$\forall x(Fx \to Gx)$	(Prem)
(3′)	Fa	($\forall$E 1)
(4′)	. . .	

But again hold on! The rationale for line (3′) in the formal proof is that we are temporarily dubbing some arbitrary element of the domain with a dummy name, just as in (3). Suppose that the domain were empty, though. We then couldn't do this.

So: in their handling of dummy names, our (entirely standard) QL inference rules presuppose that we are dealing with non-empty domains. In §27.3 and again in §28.7 we

stipulated that domains of quantification for QL languages are always to be non-empty. We now see that this stipulation is a significant one: it ties in with the natural deduction rules which we have adopted.

34.2 Two more proofs

Let's consider two more examples which illustrate the same tie between our current rules of inference and the requirement that domains are populated.

(a) Tachyons are, by definition, physical particles which are superluminal, i.e. which travel faster than the speed of light. So, adopting a QL language quantifying over physical particles, and with the obvious interpretations of the predicates, the following is true:

(1) $\forall x(Tx \rightarrow Sx)$.

Now, given that truth, we *can't* use our QL inference rules to deduce

(2) $\exists x(Tx \wedge Sx)$.

Which is just as it should be: see §29.2. We don't want the truth (1) – *if* something is a tachyon, it is superluminal – to entail a proposition that tells us that there actually *are* some superluminal particles.

So far so good. But what happens if – perhaps in the spirit of Quine's maxim of shallow analysis – we think along the following lines? The current topic is just tachyons, so we can keep things simple by taking our domain for now to be just them (compare our handling of argument **F** in the previous chapter where we kept things simple by taking the domain to be just students in the logic class). Then, using a formal language where the quantifiers run over just tachyons, the claim that tachyons are superluminal can be regimented as the simple truth

(1′) $\forall x\, Sx$.

And now we *do* have the following natural deduction proof **B**:

(1′)	$\forall x\, Sx$	(Prem)
(2′)	Sa	($\forall$E 1)
(3′)	$\exists x\, Sx$	($\exists$I 2)

But (3) is an existential claim, only true if there *is* something which travels faster than light. So, lo and behold, we now seem to have deduced the existence of something that travels faster than light from a reformulation of what was originally supposed to be a premiss that is true by definition. If only physics were so easy!

What has happened here? Again, at the second step of the proof, we take an arbitrary member of the current domain and dub it 'a'. But, as before, we can only do *that* if the domain is populated and contains some object(s) – at least one – to choose from. So, in (i) taking the domain our the relevant language to comprise just tachyons, but also (ii) assuming that we can use our current logical apparatus, we are already implicitly assuming (iii) that there *are* some superluminal particles.

(b) Suppose the domain of the relevant QL language is empty. Then, on the one hand, any wff of the language of the form $\forall\xi\alpha(\xi)$ is true (for it is always true that, *if* a thing

is in the domain, then it satisfies the condition expressed by α, since the antecendent of this generalized conditional is always false). On the other hand, in an empty domain, $\exists\xi\alpha(\xi)$ is trivially false. So, applied in an empty domain, the pattern of reasoning in the last proof will take us from truth to falsehood.

And indeed **B** *does* take us from truth to falsehood, since there are no such things as tachyons and so the stipulated domain of quantification is empty.

(c) For our second example, consider another mini-proof, **C**:

(1)		
(2)	$(\mathsf{Fa} \vee \neg\mathsf{Fa})$	(LEM)
(3)	$\exists\mathsf{x}(\mathsf{Fx} \vee \neg\mathsf{Fx})$	($\exists$I 2)

Our rules allow us to prove any instance of the Law of Excluded Middle ($\alpha \vee \neg\alpha$) from no premisses. Assume we are dealing with a QL language which has the unary predicate 'F' (expressing the property F): then in particular we will be able to prove (2) from no premisses. And then (3) follows by ($\exists$I). However (3) is an existential claim; it tells us that there is something in the domain which is either F or not F. And that can only be true if there *is* something in the domain; (3) will be false if the domain is *tachyons*!

34.3 Preserving standard logic

(a) Proof **B** illustrates that that derivations in our natural deduction system are not always truth-preserving, if we allow empty domains. Proof **C** illustrates that theorems of our system are not always true, if we allow empty domains.

Is this Bad News for our standard logic? Some logicians would say so:

> An inference is *logically* valid if it is necessarily truth-preserving in virtue of topic-neutral features of its structure (see §6.2). And *formal* logic is the study of logical validity, using regimented languages to enable us to bring out how arguments of certain forms are valid irrespective of their subject-matter.
>
> Now, sometimes we want to argue logically about the properties of things which we are already know to exist (electrons, say). Other times we can want to argue in an exploratory way, in ignorance of whether what we are talking about exists (superstrings, perhaps). While sometimes can want to argue about things that we believe don't exist, precisely in order to try to show that they don't exist (tachyons, perhaps). And we presumably want to formally regiment correct forms of inference which we can apply neutrally across these different cases. Hence *one* way our formal logic should be topic neutral is by allowing empty domains. Hence our current QL rules – being incorrect for empty domains – are not topic neutral, don't correctly capture only logical validities and logic truths, and so need revision.

Persuaded by such reasoning, some logicians indeed advocate the general adoption of a somewhat more complicated theory of *free* logic – i.e. a logic free of existence assumptions, allowing empty domains (and often also allowing 'empty proper names' which don't have a reference, though that is strictly speaking another story).

How might the defender of our standard QL logic reply?

There is no One True Logic. Choosing a formal logic involves – as we have said before – weighing up costs and benefits. And the small benefit of having a logic whose inferential principles hold in empty domains is not worth the cost. For when we want to argue about things that do not or might not exist we already have two options while using our standard logic.

First, a suitably inclusive domain is very often easily found (and indeed will typically be in play when engaged in serious enquiry rather than just running up artificial examples in the logic classroom). For example, instead of taking the domain to be *tachyons* and regimenting the proposition 'all tachyons are superluminal' as '$\forall x\, Sx$', we take the domain more naturally and more inclusively to be, say, physical particles. We then regiment that proposition as we initially did, as '$\forall x(Tx \rightarrow Sx)$' and lose the unwanted inference to '$\exists x\, Sx$'.

Second, when we do have lingering doubts about some wider domain, we can (and do) proceed in a more exploratory, non-committal, suppositional, mode. For example, if we are sceptical about sets, we might bracket our set-theoretic investigations with an unspoken 'Ok, let's take it, for the sake of argument, that there *is* this wildly infinitary universe that standard set theory talks about . . . '. And then, within the scope of that bracketing assumption, we plunge in and quantify over sets in the usual way, and continue our mathematical explorations *as if* we are dealing with a suitably populated domain, to see where our investigations get to. So yes, once we have made the supposition for the sake of further exploration that there are sets or superstrings or whatever, we might want the same logic to apply in each case, topic-neutrally. But we don't require the logic we use, within the scope of the bracketing supposition that we *are* talking about something, to still be neutral about whether there is anything in the domain.

The debate will continue. But we have perhaps said enough to explain why, at least for our introductory purposes, we can stick with with our standard logic, the logic for reasoning about non-empty domains.

34.4 Summary

Our standard natural deduction rules presuppose that we are dealing with non-empty domains.

We could revise our logic to give rules for reasoning that apply to populated and empty domains alike. But given that we almost always reason while presupposing – if only for the sake of argument – that we are indeed talking about something, the standard rules apply widely enough to warrant continuing to explore them.