# Notes

## QL languages

- Here are a group of five draft chapters from the planned second edition of *IFL* – Chs 25 to 29 – motivating and introducing QL languages.
- These chapters basically replace Chapters 21 to 24 of the first edition, keeping very much to the same approach, though heavily revised and rewritten. They are sandwiched between two short Interludes, which say where we have been and were we are going.
- 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 in the overall structure of the book.
- Since these *are* heavily revised chapters, all comments/corrections 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.

 Comments to ps218@cam.ac.uk

# An Introduction to Formal Logic

*Second edition*

Peter Smith

April 13, 2019

# Contents

# Interlude: Formalizing general propositions

(a) Since the previous Interlude, we have developed a Fitch-style framework for regimenting step-by-step proofs which rely on (tidied-up versions of) the sentential connectives 'and', 'or', 'not' and 'if'. We eventually codified the resulting PL natural deduction system in §24.2, and we need not summarize it again here.

So we have now approached questions of PL validity in two ways. First, we defined tautological validity and gave a direct brute-force method for determining validity in this sense. Then we set down some intuitively correct truth-preserving modes of inference for suitably tidied-up connectives, and codified these into a natural deduction system for warranting inferences. We can then show that — by soundness and completeness theorems – that the two approaches validate just the same inferences.

(b) But now, after all our work on propositional logic, we must at last move on. So where next?

Consider for example 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 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 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 spend some time motivate and explaining 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. Later, we will be adding *dummy names*, the *identity predicate* and *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', '$\pi$', 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* – as in 'this spoon', 'that guy', or simply 'she' (relying on context to fix the intended reference).

(iii) So-called *definite descriptions* of the form 'the $F$' – as in '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 $\pi$', '$\pi/2$', '$\sin(\pi/2)$', ....

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

(v) *Dummy names* – as in the lawyer's use of 'John Doe' and 'Jane Roe' in 'Suppose Jane Roe is the executor of John Doe's will, ...', or 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 later 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 add expressions for functions that allow us 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 Alps', '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

(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 logicians see things differently. First, we 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, we will generalize the school-room notion of a predicate. For us, a predicate 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 be added to get a sentence.

So: the ordinary-language predicates '① is a cat', '① is scary', '① mews', '① likes fish' (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, though, 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′) '$\mathsf{P}$' and '$\mathsf{Q}$' entail '$(\mathsf{P} \wedge \mathsf{Q})$',
(2′) '$\mathsf{P}$', '$\mathsf{Q}$' and '$\mathsf{R}$' entail '$((\mathsf{P} \wedge \mathsf{Q}) \wedge \mathsf{R})$',
(3′) '$\mathsf{P}$', '$\mathsf{Q}$', '$\mathsf{R}$' and '$\mathsf{S}$' entail '$(((\mathsf{P} \wedge \mathsf{Q}) \wedge \mathsf{R}) \wedge \mathsf{S})$',

and so on. Do such examples show that some ordinary-language predicates like 'cohabit' and 'entail' can combine with *varying* numbers of singular terms (for people, 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; in particular, each combines with a *fixed* number of terms to make an atomic wff.

(d) 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. Likewise, 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.

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 disjunctive or gerrymandered, is countenanced.

(e) 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 the ordered pair of $o_1$ and $o_2$ satisfies the predicate.

After ordered pairs come (of course) 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 – and we can take a 1-tuple to be an object by itself. Hence, generalizing:

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

(f) Finally in this section, we need to complicate the story just a bit, and refine our account of the arity of a predicate.

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 – so, 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'. Thus we can say that '① killed ①' takes *one* name twice over. And since, in this sense, it takes one name, we will count it as a *unary* predicate. Which seems the right thing to say, since '① killed ①' comes to the same as '① killed him/herself': it expresses a condition satisfied by *single* things (not by pairs).

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). 'Seneca killed Seneca' counts as a way of filling out the binary predicate '① killed ②' alongside 'Nero killed Poppaea'.)

## 25.3 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) Generalizing to cover the case of predicates of other arities is straightforward. The sense of a $k$-ary predicate is that aspect of its meaning which fixes the condition which a $k$-tuple of objects must meet in order to satisfy the predicate. And crucially:

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, etc.).

We will from now on 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'.)

## 25.4 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?

## 25.5 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 determines the truth-conditions of basic sentences involving that predicate. Similarly, whether or not names have senses, it is the reference of a name which determines the truth-conditions 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. Note again, in the general case, it is references and extensions which fix truth conditions for basic sentences built up from some name(s) and a predicate.

What goes for these basic sentences goes too 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! So (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. Hence, 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.

Similarly, 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 similarly 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.

Arguably, then, if a broadly Fregean account is right, senses *do* have to enter the picture to account for the truth conditions of claims like (1) to (4), where a simpler sentence is embedded inside the context '*n* believes that . . . '. However, looking ahead, *our* QL *languages will not have complex contexts like these.* In the relevant way, QL sentences will all be simple enough for *their* truth-conditions to be fixed by references and extensions alone. More about this in due course.

(c) In short, then, even granted that there are Fregean senses, they *won't* enter the story about the truth-conditions of QL sentences. But this means that they won't enter either into the account of what makes for necessarily truth-preserving arguments involving such sentences. That is why we needn't worry any more about the theory of sense. We should be very grateful for small mercies!

## 25.6 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, but contentiously, 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 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 $G$s,
> There are infinitely many $G$s.

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. So 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 identity 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 modifying tenses 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 (8) and likewise (11) is unambiguous, one way or the other, 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*. (A name like 'Socrates' and a generalizing expression like 'someone' can both traditionally count as subject terms – but *here* is a deep reason not to want to classify together.)

(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 complex 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 need contextual clues and interpretative charity 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

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, we saw in the last chapter that we can in fact usually do the trick 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., which 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 an instance of 'Every *F* is *G*', where *F* is replaced by the colourless, all-inclusive, 'thing'. Similarly, for example, 'Somebody is *G*' is an instance 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 – trade in 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 and their associated variables 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. In other words,

> We fix at the outset, once and for all, by a description $D$ in the glossary for the language, one common domain for the quantifiers of a particular QL language.

(b) This simple convention buys us clarity and simplicity – though at the cost of some artificiality, including some departure from mathematical practice.

For note, 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 different sorts of variables to range over different 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, *students signed up to the logic class*. Or they can be very large – as when our quantifiers range over *everyone now living*, or over all *natural numbers* or all *elementary particles*.

Could a domain be *empty*, i.e. can we quantify over nothing at all? Or at the other extreme, can 'everything' (for example) range over, well, *everything* – meaning not just things of this or that specific kind but absolutely every object that exists?

Later, we will follow convention and ban empty domains (see §**??**). However, we can allow a language's universe of discourse to be absolutely everything, *if* that idea makes sense. But 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, 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) $(\forall x)$(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

(3) $(\forall x)(x \text{ is in the class} \rightarrow x \text{ has arrived})$.

The next sentence, however, is potentially ambiguous

(4) Everyone in the class has not arrived.

We can, in some contexts, understand this with 'not' having wide scope, i.e. we can understand (4) as conveying the message unambiguously expressed by

(5) $\neg(\forall x)(x \text{ is in the class} \rightarrow x \text{ has arrived})$.

But in other contexts, we will understand (4) as meaning

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

The relative scopes of the negation and quantifier are now made transparently clear.

Likewise,

(7) Some footballer deserves great riches

can be partially symbolized as

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

or equivalently as

(9) $(\exists x)(x \text{ is a footballer} \wedge x \text{ deserves great riches})$.

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

(10) Some footballer does not deserve great riches.

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

(11) $(\exists x)(x \text{ is a footballer} \wedge \neg\, x \text{ deserves great riches})$,

(12) $\neg(\exists x)(x \text{ is a footballer} \wedge x \text{ deserves great riches})$.

Again, the quantifier operator and the negation are thus very perspicuously revealed as having different scopes in the two readings.

## 27.5 Loglish

(a) The unholy mixture of English and formal logical symbolism that we have fallen into using is often called *Loglish*. And as we will see in the coming chapters, it is *very* handy for easing the transition between ordinary-language and our official fully formalized QL languages to mix English and formal devices like this in various degrees.

When we turn to defining our formal languages, we will eventually give proper syntactic rules for forming wffs with quantifier prefixes and also give semantic rules

for interpreting quantified wffs. But it will help to prepare the ground if we now give some informal rules which fit our intuitive understanding of Loglish quantifiers, and then these rules can be carried over in giving an introduction to QL languages.

(b) The syntax is easy! In headline terms: we form a quantified sentence by starting from a sentence involving one or more occurrences of a particular name, swapping the name for a variable, and prefixing a quantifier using the same variable. We just need to ensure we pick a variable not already in the sentence – we must not get entangled with any existing ties between quantifiers and variables. So the informal rule is:

> Suppose $A(n)$ is a Loglish sentence involving one or more occurrences of the name $n$. Let $A(v)$ be the result of replacing the name $n$ throughout that sentence by a variable $v$ new to the sentence. Then $(\forall v)A(v)$ and $(\exists v)A(v)$ will be also be Loglish sentences.

We can then say that $(\forall v)A(v)$ and $(\exists v)A(v)$ are formed by *quantifying into* the place(s) occupied by $n$ in $A(n)$.

For example, starting from 'Romeo loves Juliet' we can grammatically form the Loglish sentence '$(\exists y)$ Romeo loves $y$'. And from that we can form '$(\forall x)(\exists y)$ $x$ loves $y$' (or equally, '$(\forall z)(\exists y)$ $z$ loves $y$', etc.) by quantifying into the place occupied by 'Romeo'. Similarly, starting from the Loglish '(David is a footballer $\wedge$ $\neg$ David deserves great riches)' we can form the sentence '$(\exists x)$($x$ is a footballer $\wedge$ $\neg$ $x$ deserves great riches)'.

(c) To settle the meaning of quantified Loglish sentences we have to specify what the quantifiers are ranging over. Assume that's been done – i.e. we have given a description of the intended domain of quantification. And we will assume too that as far as the name $n$ is concerned, what matters for the truth or falsity of $A(n)$ is the reference of $n$ (so we are not dealing with one of those more complex cases touched on in §25.5).

Then $A(n)$ can be read as saying that the object referred to by the name $n$ satisfies some condition expressed by the rest of the sentence – it says, let's suppose, that the object is $C$. Then the informal semantic rule for the related quantified sentences is:

> $(\forall v)A(v)$ says that everything (in the relevant domain) is $C$.
> $(\exists v)A(v)$ says that at least one thing (in the relevant domain) is $C$.

Note: even once we have fixed the condition expressed by $A$, the content of the claim $(\forall v)A(v)$ will vary depending on how we have specified the domain of quantification.

(d) What holds of everything in a domain holds for any particular named thing in the domain – so from $(\forall v)A(v)$ we can infer a corresponding instance $A(n)$. (We assume names do refer to things in the domain of quantification!)

*But the converse is false*. It could be that for every 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 by the fact that just the *named* stars happen to contain iron.

Similarly, a sentence $(\exists v)A(v)$ is true if and only if something in the relevant domain

satisfies the condition expressed by $A$. And since what holds of a particular named thing holds of something, 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 converse is false. It could be that $(\exists v)A(v)$ is true, even if each 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 $y$, ...

Here is a variant notation which ties quantifier prefixes to places graphically. Take the predicate '① loves ②'. Then instead of (1) or (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(f)). 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 $y$,

(4) ∀ ∃ 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 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 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 introduced

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

## 28.1 Names, predicates and atomic wffs in QL

(a) The headline news is that 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). However, the only terms we are initially going to be introducing are *proper names*. Which means that, 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 there is an entirely standard convention of using (mid-alphabet) *lower case* letters for proper names and *upper case* letters for predicates – already 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. So:

> 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 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, . . ..

Note, by the way, terms can be repeated.

This way of structuring atomic wffs is, however, 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 when convenient; we will also allow the use of non-letter symbols e.g. for names and/or predicates.

(c) Now for semantics. We again follow the lead of our discussion of ordinary-languages names and predicates in Chapter 25. So QL names will refer to objects (in that broadly inclusive sense of 'object' that can include people and mountains, stars and molecules, numbers and sets, etc.). QL predicates will 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, as you would have predicted,

> We interpret a QL proper name for an object by associating it with an (in context, unambiguous) English name or other expression referring to some unique object.
>
> We interpret 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.

(d) Now for a rather tricksy 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 whether we say that a QL language (i) can have predicates of various arities from one up, plus propositional letters, or (ii) can have predicates of various arities from zero up. The second way of putting it is just neater!

(e) We need to add a very important proviso. We want the sentences of QL languages to be like the wffs of PL languages – either determinately true or determinately false in any situation (see §10.3). This determinacy assumption will play a central role in our account of q-validity for QL inferences just as it did in our account of tautological validity for PL inferences.

In particular, then, this means that the condition expressed by a predicate has to be a *sharp* one that doesn't allow borderline cases – in other words, it is always a determinate matter whether or not the predicate is satisfied by some given object(s). Therefore, when we interpret a formal predicate by associating it with an English predicate, we will have to quietly assume that any vagueness in the English has been tidied away or can just be ignored.

## 28.2 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) 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.

## 28.3 Adding the connectives

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

> If $\alpha$ and $\beta$ are wffs of a given QL language, so too are $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, and $(\alpha \to \beta)$. '$\bot$' 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 again expresses a non-specific contradiction.

Given these newly added rules for wff-building with connectives, the following are therefore also wffs of $\mathsf{QL}_1$, with the associated interpretations:

$\neg\mathsf{Fm}$: Socrates is not a philosopher.
$(\mathsf{Go} \rightarrow \mathsf{Ho})$: If Aristotle is a logician, he is wise.
$(\mathsf{Hn} \wedge \mathsf{Lmn})$: Plato is wise and Socrates loves him.
$(\mathsf{Lom} \wedge \neg\mathsf{Lno})$: Aristotle loves Socrates but Plato doesn't love Aristotle.
$(\mathsf{Rnmo} \rightarrow (\mathsf{Lom} \wedge \neg\mathsf{Lon}))$: 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 – when we construct a wff stage-by-stage, a connective gets introduced as connecting *wffs*. So this means that we must translate e.g. 'Plato and Aristotle are philosophers' into $\mathsf{QL}_1$ as

$(\mathsf{Fn} \wedge \mathsf{Fo})$

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

$\mathsf{F}(\mathsf{n} \wedge \mathsf{o})$.

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 translate 'Aristotle is a logician and is wise' into $\mathsf{QL}_1$, we must use

$(\mathsf{Go} \wedge \mathsf{Ho})$

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

$(\mathsf{G} \wedge \mathsf{H})\mathsf{o}$.

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

## 28.4 Syntax for the quantifiers

(a) Following up the proposals in §27.1 and §27.4, we now add to QL languages the quantifier symbols and give ourselves a supply of variables to form quantifier prefixes and linked pronouns. Thus:

Every QL language has an unlimited supply of *variables*, starting

$\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots$ .

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, 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 the modern one, 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 variables are to be sharply distinguished from proper names (and other terms).

(b) So how do we form quantified QL wffs? Remember the informal Loglish idea: we can form a quantified sentence by quantifying into places occupied by some name(s). In other words, starting from a sentence involving one or more occurrences of a particular name, we swap out the name for a variable new to the sentence, prefix a quantifier using the same variable, and we get a new quantified sentence.

We now transpose this idea to give us an outline story about quantifier syntax in QL. First we adopt a couple of conventions for the use of schematic letters:

'$\xi$', '$\zeta$' – *xi*, *zeta*, the Greek *x*, *z* – will be used to schematically represent formal variables (the natural choice, at least given our general policy of using Greek letters in schemas representing formal expressions!).

'$\tau$' – *tau*, the Greek *t* – will be used to schematically represent a term.

For the moment, the only terms we have in play are proper names; but we will future-proof our QL construction rule by giving it in a general version:

Suppose $\alpha(\tau)$ represents a QL wff involving one or more occurrences of the term $\tau$. Then $\alpha(\xi)$ represents the expression which results from replacing the term $\tau$ throughout $\alpha(\tau)$ by a variable $\xi$ new to that wff.

Given $\alpha(\tau)$ is a wff, then $\forall\xi\alpha(\xi)$ and $\exists\xi\alpha(\xi)$ will be also be QL wffs.

Note however that the expression $\alpha(\xi)$ formed from the wff $\alpha(\tau)$ is not, repeat *not*, a wff for us. We don't get a wff until we prefix a matching quantifier.

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

(1) $(Fm \to Hm)$.

Now replace the term (in this case, name) 'm' by the variable 'x' and prefix the result with '$\forall$x'. We get the wff

(2) $\forall x(Fx \to Hx)$.

Equally, starting from

(3) $(Fm \wedge Lmn)$

and applying our construction rule, we can get to e.g. the wff

(4) $\exists z(Fz \wedge Lzn)$.

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 '$\exists x$'. Which gives us the wff

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

Similarly, starting from the $QL_1$ wff

(6) $Lmn$

we can construct in turn

(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 however that our construction rule doesn't allow 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 into the places held by 'm' in

(13) $Lmm$

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

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

Then, quantifying into the position now held by the name 'm', we get e.g.

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

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

(16) $Fz$, $Lzn$, $Myx$, $Rxoy$

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

And, given our insistence that we can't reuse variables already appearing in a given wff when applying a quantifier, we *can't* get from (7) to the expression

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

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

(18) ∀z∀x∃y Lxy.

Neither of the expressions (17) and (18) will count as wffs.

(e) We will need to return to tidy up our account of QL syntax later: however, our story so far will serve as an easily-grasped introduction. A quick reality check, though. We just said that 'Fz' and 'Lzn' are *not* wffs. So inside the wff (4) '∃z(Fz ∧ Lzn)' the connective '∧' does *not* connect wffs. Still, '∃z(Fz ∧ Lzn)' is constructed by quantifying a wff like (3) '(Fm ∧ Lmn)'. And *there* in (3) the connective *does* connect wffs. Generalizing, as we put it before, in building up a wff, a connective does indeed first get introduced as connecting whole wffs. That's why we can say that connectives in a QL language do remain essentially sentential connectives.

## 28.5 Interpreting the quantifiers

(a) Recall from §27.5(c) how we interpret the quantified sentences of Loglish. We first have to specify a domain of quantification. Then suppose that the sentence $A(n)$ says that the object referred to by $n$ satisfies some condition – suppose it says that the object is $C$. Then

$(\forall v)A(v)$ says that everything (in the relevant domain) is $C$.
$(\exists v)A(v)$ says that at least one thing (in the relevant domain) is $C$.

We now follow this model in interpreting our formal wffs.

So, first step:

> To interpret a QL language we must specify its universe of discourse. We do this in the glossary for the language by giving some *domain description*.

The domain for a QL language can be any set of objects, large or small – with one proviso: the objects named by the proper names of the language should be in the domain of quantification. Why so? Because, in a given framework, we want any named object to be covered by 'everything' – so that what holds true of everything in the domain will hold true of any particular named thing.

(b) With the domain now fixed, we can continue in parallel to the story about Loglish:

> Suppose that, in a given QL language, the term $\tau$ 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 $C$.
>
> Then $\forall\xi\alpha(\xi)$ says that everything is $C$ – meaning everything in the relevant language's universe of discourse, of course.
>
> Similarly, $\exists\xi\alpha(\xi)$ says that something, at least one thing, is $C$ – meaning something in the relevant universe of discourse again.

(c) We will illustrate this by working through some examples from our mini-language $\mathsf{QL}_1$. But first we need to complete the interpretation key for this language by fixing what its quantifiers range over. So we stipulate:

The domain of quantification for $\mathsf{QL}_1$: people, past and present.

Let's start off with the wff

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

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

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 again, 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) $\mathsf{Lmn}$.

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.

Of course, we don't in practice have to build up to the interpretation of (7) like this! We can just directly transcribe (7) into Loglish as

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

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

(d) Let's have another series of examples.

(9) $(\mathsf{Fm} \wedge \mathsf{Romn})$

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

(10) $\exists \mathsf{y}(\mathsf{Fy} \wedge \mathsf{Royn})$?

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) $(\mathsf{Lo} \rightarrow \exists \mathsf{y}(\mathsf{Fy} \wedge \mathsf{Royn}))$

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) $\forall \mathsf{x}(\mathsf{Lx} \rightarrow \exists \mathsf{y}(\mathsf{Fy} \wedge \mathsf{Rxyn}))$.

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 (11). Again, the thing to do is just transcribe it into Loglish as a half-way house, as in

(13) $(\forall x)$(if $x$ is a logician, then $(\exists y)$($y$ 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 chapter.

## 28.6 Quantifier equivalences

(a) Given our glossary for $\mathsf{QL}_1$, and the rule for interpreting quantified wffs,

(1) $\forall \mathsf{x} \neg \mathsf{Fx}$

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

(2) $\neg \forall \mathsf{x} \neg \mathsf{Fx}$

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) $\exists \mathsf{x} \mathsf{Fx}$.

Similarly, since

(4) $\exists \mathsf{x} \neg \mathsf{Fx}$

in $\mathsf{QL}_1$ means that someone is not a philosopher, its negation

(5) $\neg \exists \mathsf{x} \neg \mathsf{Fx}$

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) $\forall \mathsf{x} \mathsf{Fx}$.

And the two equivalences here obviously generalize. Use '$\xi$' as before to indicate a variable, and now use '$\varphi$' (*phi*) for some suitable wff-completing expression, then

In QL languages, wffs of the forms $\neg\forall\xi\neg\varphi$ and $\exists\xi\varphi$ (with the same completing expression $\varphi$) 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. 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 '$\exists$xFx' (for example) as in effect a big disjunction like '$\mathsf{Fm} \vee \mathsf{Fn} \vee \mathsf{Fo} \vee \ldots$'. By a version of one of De Morgan's Laws, this big disjunction is equivalent to the *negation* of the big conjunction '$\neg\mathsf{Fm} \wedge \neg\mathsf{Fn} \wedge \neg\mathsf{Fo} \wedge \ldots$' (why?). Given the same pretence that everything has a name, that big conjunction is in effect equivalent to '$\forall$x$\neg$Fx'. Whence, just as we want, '$\exists$xFx' is equivalent to the negation of that, i.e. '$\neg\forall$x$\neg$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 (compare §13.4). 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 $\mathsf{QL}_1$ wffs '$\forall$x$\exists$yLxy' and '$\exists$y$\forall$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 before moving on. Thus consider

(1) $\forall$x$\forall$yLxy

which says that everyone has the property which '$\forall$yLmy' which attributes to Socrates, i.e. the property of loving everyone. So (1) says that everyone loves everyone. While, swapping round the quantifiers,

(2) $\forall$y$\forall$xLxy

says that everyone has the property which '$\forall$xLxn' which 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) $\exists$x$\exists$yLxy
(4) $\exists$y$\exists$xLxy

are also equivalent, both saying that someone loves someone.

The point generalizes. Suppose $\xi$ and $\zeta$ 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.

## 28.7 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 fix 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 '∀x', '∀y', . . . , are now available as universal quantifiers: expressions like '∃x', '∃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 $\tau$, replacing the term throughout by a variable $\xi$ new to the wff, 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 $\tau$ 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.

## Exercises 28

# 29 Translations

In Chapter 10, after talking about the *interpretations* of PL wffs (which assign them senses, i.e. truth-conditions), we quickly moved on to talking about *valuations* (assigning wffs truth-values) – and it is valuations that are involved in the definition of tautological validity for PL arguments.

There is a similar double story to be told about the semantics of QL languages. We now know something about how to *interpret* the wffs of such a language (i.e. how to assign them senses, i.e. truth-conditions). But to give an account of truth-preserving QL arguments, we will need to move on to give an account of how to assign wffs truth-values by what we will call a *q-valuation*. And we will then be able to define a corresponding notion of q-validity for QL arguments.

However, this story will have to wait. First we need to get more experience at translating as best we can between English and QL. That's the business of this chapter.

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 nearly all 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 '∀x(Fx : Gx)' or '(∀x: Fx) Gx' – to say that all *F*s 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 $\mathsf{QL}_1$ 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 $\mathsf{QL}_1$ 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 '$\exists x(Gx \to Hx)$' would also be wrong. For take some non-logician; Socrates will do. Then (Socrates is a logician → Socrates is wise) is *true*, being a material conditional with a false antecedent. Hence there *is* someone $x$ such that ($x$ is a logician → $x$ is wise). Hence the mere existence of a non-logician like Socrates makes '$\exists x(Gx \to Hx)$' true. But obviously that's not enough to make it true that some logicians are wise.

Further, we want the translations of intuitively equivalent ordinary-language propositions to come out as 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 $\mathsf{QL}_1$ translations should be equivalent.

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

So putting things together, 'Not all logicians are unwise' is rendered as '$\neg\forall x(Gx \to \neg Hx)$' 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) $\forall x(Gx \to Hx)$.

(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) $\forall x(Gx \to Hx)$,
(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 $G \to x$ 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 '$\forall x(Gx \to Hx)$', 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 we can render that prefix by either of the corresponding formal expressions '$\forall x\neg$' or '$\neg\exists x$'. Or putting that rather more carefully, with the same notation as in §28.5,

> Suppose we interpret the term $\tau$ 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 $C$s (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 *F*s' by '$NxFx$'. 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 formalizes the thought that everyone is such that, if a philosopher, then not wise:

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

The second option formalizes the equivalent thought that there doesn't exist someone who is a philosopher and wise:

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

Why are these equivalent? Because $\forall x(Fx \to \neg Hx)$ 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 via Loglish (relying on the key fact that the meaning of the QL and the Loglish quantifiers is, by design, the same.)

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 '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* Assuming we have fixed what/who our unrestricted generic quantifiers range over, replace the informal restricted quantifiers by using these generic quantifiers 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 which are still straightforwardly interpretable, relying on 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 isn't being suggested that you have to follow this stage-by-stage plan in all its detail, or that you have to do things in the exact order suggested. You will 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) Maldwin 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(Lxm $\wedge$ (Lxn $\wedge$ 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 $\mathsf{QL}_2$'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. We will see a similar example in a moment.

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) Maldwin is married to no one
- ≃ It is not the case that Maldwin is married to someone

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

Or, starting again, taking the alternative route:

≃ (Everyone $x$ is such that) Maldwin 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)
≃ $\neg\exists x(\mathsf{L}\mathsf{n}x \wedge \mathsf{L}x\mathsf{o})$.

Or alternatively

≃ Everyone who loves Nerys does not love Owen
≃ (Everyone $x$ who loves Nerys is such that) $x$ doesn't love Owen
≃ (Everyone $x$ is such that)(if $x$ loves Nerys then $x$ doesn't love Owen)
≃ $\forall x(\mathsf{L}\mathsf{n}x \rightarrow \neg\mathsf{L}x\mathsf{o})$.

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

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

Alternatively, we have

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

We will say more about why these are equivalent in §29.8.

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)
≃ $(\forall x\mathsf{L}\mathsf{n}x \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$.

Similarly,

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

But now what about 'If anyone loves Nerys then Owen does'? Perhaps with the right stress and/or right context we can read this as 'If just *anyone* loves Nerys, then . . . ' which is equivalent to (10). However, the 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
≃ $\forall x(\mathsf{L}\mathsf{n}x \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$.

However, you might ask, 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 '$\forall x(\mathsf{L}\mathsf{n}x \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$' and '$(\exists x\mathsf{L}\mathsf{n}x \rightarrow \mathsf{L}\mathsf{o}\mathsf{n})$' are in fact equivalent – see §29.8.

Suggestion: pause now to tackle the first, easier, set of Exercises.

## 29.6 More translations into $QL_2$

(a) Now for some more examples of translating claims about Welsh affairs which involve sentences with 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.

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

An equivalent translation is also available: '$\forall x \exists y(Fx \rightarrow Lxy)$'. (See §29.8.)

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

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

(4) No woman loves every man
  ≃ It is not the case that some woman loves every man
  ≃ ¬ Some woman $x$ is such that $x$ loves every man

$\simeq$ ¬(Someone $x$ is such that)($x$ is a woman and [$x$ loves every man]
$\simeq$ ¬(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 ¬[$x$ loves every man])
$\simeq$ (Every $x$ is such that)($x$ is a woman $\rightarrow$ ¬[(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 '$\forall x(Gx \rightarrow \neg\exists y(Fy \wedge Lxy))$'. 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 again involves translating the unary predicate 'is married' into a language which only has a built-in binary predicate meaning 'is married to'. No problem! – we just rely on the obvious semantic equivalence of 'is married' to 'is married to someone'. Thus we have:

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

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

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

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

(7) A married man only loves women
- $\simeq$ (Every man $x$ who is married is such that)($x$ only loves women)
- $\simeq$ (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,

- $\simeq$ (Everyone $x$ is such that)([$x$ is a man and is married] $\rightarrow$ [(everyone $y$ is such that) if $x$ loves $y$ then $y$ is a woman])
- $\simeq$ $\forall x((Fx \wedge \exists z Mxz) \rightarrow \forall y(Lxy \rightarrow Gy))$.

## 29.7 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 '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 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) $\neg\forall x\forall y((Fx \wedge Gy) \rightarrow (Mxy \rightarrow \exists z Rzxy))$
- $\simeq$ $\neg$(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)
- $\simeq$ $\neg$(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 $\exists y Rxmy)$

≃ If no woman loves Maldwyn then, for any $x$, $x$ loves Maldwyn if and only if $x$ is a child of Maldwyn and someone

≃ If no woman loves Maldwyn, then he is loved just by any children he has.

## 29.8 Moving quantifiers

(a) Let's comment on a couple of general points raised by some of our examples. Consider (9) from §29.5, '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 (think about it!).

This 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 $\beta$ doesn't contain the variable $\xi$. But this 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 $\forall \xi(\alpha(\xi) \wedge \beta)$
$(\exists \xi \alpha(\xi) \vee \beta)$ and $\exists \xi(\alpha(\xi) \vee \beta)$

where $\beta$ doesn't contain the variable $\xi$. 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.

$(\exists x Lxn \rightarrow Lon)$

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

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

which by the interrelation of the quantifiers is equivalent to

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

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

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

which is equivalent to

$\forall x(Lxn \rightarrow Lon)$

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 are equivalent:

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

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 $\beta$ doesn't contain the variable $\xi$.

You have to keep your wits about you when dealing with conditionals!

## 29.9 Summary

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

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 $F$s' by the corresponding wff '$\neg\exists xFx$' or equivalently '$\forall x\neg Fx$'.

Translating from English into a QL language can be thought of as in effect taking five main stages, having fixed what the quantifiers range over:

(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 '$\forall$' and '$\exists$' 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.

The reverse journey takes us back from QL wffs to their English translations.

## Exercises 29

# Interlude: Arguing in QL

(a) We said in the previous Interlude that we are going to 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 – QL languages. And then we assess the resulting formalized arguments.

We have now made a start on characterizing QL languages (though there are further ingredients we haven't yet introduced). In particular, we have met the crucial Fregean idea of using a quantifier-variable notation to regiment general propositions, and then seen how to formally implement this idea.

So we can now turn to thinking about the assessment of QL arguments.

(b) Let's recall again that we approached questions of PL validity in two ways:

(P1) We defined tautological validity for arguments involving the tidied-up 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 these connectives, and codified them into a natural deduction system for warranting inferences.

Soundness and completeness theorems tell us that two approaches can 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 natural deduction system to cope with quantificational reasoning turn out to be surprisingly straightforward: we will meet them in the next chapter. But let's pause to say just something here about the new idea of q-validity.

(c) 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 its vocabulary – i.e. every way of assigning q-values to its names and predicates, and every way of fixing the domain – if the premisses are true then the conclusion is true too.

Take a toy example, the 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

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

Again, this is intuitively valid in virtue of its quantification structure. And this idea can now be cashed out by saying that the argument is q-valid. It doesn't matter what domain the quantifiers are ranging over, it doesn't matter which object in the domain 'n' picks out, it doesn't what 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 should be true!)

(d) 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 if 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 '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 no generally applicable test.)

(e) So what other approaches are there to demonstrating 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 §16.1 (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 said, this framework can be smoothly and naturally(!) extended to give us proofs of 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*.

Therefore, starting in the next chapter, we will begin to explore a natural deduction system for quantificational arguments, initially just claiming that the rules are intuitively compelling. Only later will we return to give an official account of q-validity and explain why our rules are indeed reliable for q-validity.