Death, Taxes and Logic: 3 Certain Things

Propositions, Structures and the composition of Truth

Volume II: The Logic of Predicates and Quantifiers

By Devon "Smoove D" Belcher

Table of Contents

A Note on Conventions

I have decided to keep the conventions of NL and QL separate. If I am writing in NL, names are capitalized, and predicates are lowercase. I hope this does not cause too much confusion, especially in the case of talking about models.

Chapter I

Why Quantifiers? The Syntax of QL

Introduction

Consider the following argument:

Albert is a pirate
All pirates are sexy
∴ Albert is sexy.

How should this argument be represented?

P(a)
?
S(a)

What is that second sentence? Is 'all' a subject term, as in Sexy (allpirates)? But then we have no natural connection to albert in premise 1 and the conclusion. Is it a hidden identity, as in

P = a
P = S
a = S

That can't be right either. After all, if the first premise is a hidden identity, then it'd also be true that 'Hanna is a pirate' was an identity, and that would net us:

P = a
P = h
∴ a = h.

But Albert is not Hanna, and in general, just because two things share a predicate, that does not mean they are identical.

Another pitfall of treating 'all' or 'all pirates' as a subject, on par with an individual constant, is this. I am particularly a fan of Hendrick's gin, but when I can't get it, Gordon's is better than nothing. So

Gordon's gin is better than nothing
Nothing is better than God
Thus, Gordon's gin is better than God

Clearly, whatever one's theological proclivities, this conclusion is outrageous. Hendrick's isn't better than God, and Gordon's isn't even as good as Hendrick's. But this result comes about because we are implicitly treating 'Nothing' as a subject term, i.e., “BetterThan (Gordon's, nothing). (We're also relying on the transitivity of the better-than relation, but that seems uncontroversial.)

The moral of the story is that phrases like Something, Everything, Nothing, and so on, are considerably more complex than their surface structure indicates. This volume will be dedicated to detailing the logic of these – the 'Quantifiers'. We will use just two, Some(thing) and All (things), and then build the rest from those two. If spending a few hundred pages on two words seems the height of pedanticism, just consider that one of the founders of modern logic, Bertrand Russell, spent 30 pages on the word 'The' in one of the most famous articles of the 20th century. Let us look at a little bit of history – feel free to skip this next section if you just want to get to the logic, though.

Historical background

Aristotle is widely considered the father of logic. Everything we did in Volume I is pretty much present in Aristotle, with some minor tweaks coming from Greek and Roman philosophers after him, and then from the Medieval philosophers in the 12th and 14th centuries. (I don't mean to discount the important contributions of the Arab world, but I simply am not really familiar with that part of history). In fact, by the time Kant writes the Critique of Pure Reason in 1781, he basically claims that logic is a completed discipline. But it was not – even the Medievals struggled trying to figure out the quantifiers. It was not until 1879 that Gottlob Frege published the Begriffschrift that we get a logic of quantifiers, a development that is completely new. But Frege's original notation was so obscure and difficult to follow that it did not really catch on until Bertrand Russell discovered Frege's work and cleaned it up a bit.

Enough history. Let's do some provinating.

The Language QL

Individual Terms

Our language will look much like our original language, with some extras thrown in. But these extras will force fundamental changes in the way we provinate.

As in PL our first syntactic category is individual terms. They work just the way they did in PL, except we add one new class of individual term – the variable. By tradition, we use x, y and z (with subscripts if necessary) as our variables. Variables in QL function much like pronouns ('he', 'she', 'it') in natural language. Of course, they don't carry any information about gender or number the way English pronouns do. We will say that variables 'range over' the things referred to by individual constants, or 'take as their values' the things referred to by individual constants.

There is nothing else new here.

Predicates and Relation Terms

There is nothing new here. Note that we do *not* have variables for predicates; this costs us some expressive power, but there are reasons for it. We will discuss them in the appendix.

Atomic Formulae

Again, these work exactly the same way – so far – as they did in PL. If you have a predicate φ with an arity of n, and exactly n individual terms, you can glom the terms behind the predicate and have a well-formed formula or WFF:

Basic WFF formation rule for QL:
If φ is a relation term with an arity of n, and $\alpha_1, \ldots \alpha_n$ are exactly n individual terms, then $\varphi(\alpha_1, \ldots \alpha_n)$ is a WFF

Remember, the α's are metalanguage variables ranging over names (not things) like 'albert', 'wooster' and so forth – with the added new twist that they can also range over variables like 'x' 'y' and 'z'. So, some examples of atomic WFFS:

Pirate (albert)
Merchant (x)
Pillages (x, hanna)
Between (y, trump, cruz)
Pillages(x,x)

There is no reason all the individual terms have to be variables, or constants – there can be any mix of them, just as long as there are as many of them as the arity of the predicate or relation. There is also no reason the individual terms in an atomic wff cannot be repeated (as in Pillages(x,x)).

Connectives and Complex Boolean WFFs

These work exactly as they do in PL. If you have any two WFFs, atomic or not, you can glom them together by putting a connective in between and slapping a pair of parentheses around them (or you can negate a single WFF).

Complex Boolean WFF Formation Rule
If φ and ψ are wffs (*atomic or otherwise!*), then
$(\varphi \wedge \psi)$
$(\varphi \vee \psi)$
$(\varphi \rightarrow \psi)$
$(\varphi \leftrightarrow \psi)$
and $\neg\varphi$ (or $\neg\psi$)
are WFFs

The only difference (and it can't be seen here) is that sometimes φ and ψ will be WFFs that themselves contain quantifiers.

Examples:

(Pirate (albert) ∧ Merchant (wooster))
(Pirate (x) ∧ Merchant (albert))
((Pirate (x) ∧ Merchant (y)) → Pillages (x,y))

There's one kind of Boolean sentence I've left out. That's when φ and / or ψ themselves contain quantifiers, because we haven't shown how to form those WFFs yet. So let's do that now.

Quantifier Phrases

Now it's time to add in the quantifiers. We have two, 'some' and 'all'. Respectively, we represent these as ∃ and ∀. We could have just stuck with 'some' and 'all', but we have math envy and we like symbols. As a mnemonic, remember that 'all' goes with the upside-down A, and 'some' is really 'there exists', which goes with the backwards E. We call ∀ the universal quantifier and the ∃ the existential quantifier, for what I hope are obvious reasons. Their analogues in natural language are *determiners*.

A quantifier on its own is not a useful syntactic or semantic unit. If I said 'some', you'd be perplexed and ask 'Some what? Some pirates? Some merchants?'. Same goes for 'all'. So to turn a quantifier into a quantifier phrase, we will glom an individual term onto it. Either a constant or a variable, but, as we'll see, adding a constant is pointless. It's just easier to formulate the rules this way and then ignore quantifier phrases with constants. Some examples of Quantifier Phrases:

∃x
∀z

Quantified Atomic WFFs

At this point you may be wondering why I am talking about WFFs (pronounced "whiffs") instead of sentences. We will get to that in a minute – basically, a sentence is a special sort of WFFs. For now, the most basic quantified WFF is simply the result of taking a WFF and glomming a Quantifier phrase in front of it:

Quantified Atomic WFF Formation Rule
If φ is an atomic WFF, then
∀αφ is a WFF
(where a is a metalanguage variable ranging over individual terms such as 'albert', 'wooster', 'x', 'y', and 'z'.)

Some examples:

∀xPirate(x)
∃xMerchant(x)
∀aPillages(a, x)
∃ySailsTo(a, x)
∀xSailsTo(a, b)

WFF 3 is an odd case. There is a constant in the quantifier phrase, but it isn't doing any work, as we will see when we get to the semantics. Nonetheless, the formula is legit. Similarly for the quantifier phrases in WFFS 4 and 5 – they contain variables, but nothing else in the sentence talks about those variables, so they are kind of pointless. We call this “vacuous quantification”. Nonetheless, it too is legit.

Quantified Complex WFFs

The treatment of quantifying atomic sentences can easily be extended to more complex sentences. Basically, if you have a WFF of any sort, whether it's atomic, Boolean, or quantified, you can form a new WFF by sticking another quantifier phrase in front of it:

Quantified General WFF Formation Rule
If φ is a WFF of any sort, then
∀αφ is a WFF
(where a is a metalanguage variable ranging over individual terms such as 'albert', 'wooster', 'x', 'y', and 'z'.)

Some examples:

(the examples from the previous section)
∀x(Pirate(x) → SailsTo(x, tortuga))
∀x∃y(Pirate(x) ∨ SailsTo(x, tortuga))

Look at that last WFF. How can you tell it's a WFF? Well, it is the result of taking a quantifier phrase, ∀x, and glomming it onto a string of symbols ∃y(Pirate(x) ∨ SailsTo(x, tortuga)). So if that larger string (starting with ∃y) is a WFF, then by the WFF formation rule, the whole thing is a WFF. Is ∃y(Pirate(x) ∨ SailsTo(x, tortuga)) a WFF? It's the result of taking a quantifier phrase, ∃y, and glomming it onto another string, (Pirate(x) ∨ SailsTo(x, tortuga)). So again, if the disjunction is a legit WFF, then by the Quantified WFF formation rule, the whole thing is a WFF. And you can tell that the disjunction is a WFF by the basic Boolean rule (unchanged from volume 1 really).

Nothing else is a WFF. That's it. That's the whole syntax of QL. Let's talk about kinky stuff now.

Of Freedom and Bondage

What does a variable really represent? What do sentences with variables in them really mean? On their own – nothing, really. Remember how I said that variables were like pronouns? A sentence with just a variable in it is sort of like walking into a room and saying 'He's a pirate'. Who on earth is Smoove talking about? Sometimes this happens when we read – we come across a pronoun, and we can't figure out what it's referring to. In such cases, we usually backtrack until we find the most recent occurrence of a name that has the appropriate gender or number (i.e., if the pronoun is 'he', it probably isn't referring to Sally).[1] It's a mark of bad writing when you have to do that too often or backtrack too much. Anyway, we can think of the pronoun as being *bound* to that name (or sometimes to a description or predicate). The pronoun only has a referent in the context of that earlier name or predicate.

So it goes in Q_L. A variable in a sentence is meaningless on its own and only gets its meaning from being attached to another phrase – in our case, to a quantifier phrase. If I just said, Pirate(x), that would be akin to saying 'he is a pirate' but without pointing to anyone or indicating anyone. Anyone listening would think they had missed some earlier bit of information that let them know something about 'he'.

The first concept we need to talk about is the concept of scope. Recall in PL, negations $\neg$ had scope. Saying $\neg P \wedge \neg Q$ was quite different from $\neg(P \wedge Q)$. In the former, each the negation has scope over only the subsequent atomic sentence, that is, P is being negated, and Q is being negated. In the latter,

1 Incidentally, that's probably why language and pronouns are gendered – it makes it easier to track pronouns.

the negation has scope over the whole conjunction: $(P \wedge Q)$ is being negated. That's a weaker claim, because one of P or Q might still be true and the conjunction false, whereas, in the former, both P and Q are false. The scope of the negation is the WFF immediately following it, paying attention to parentheses. If what follows the negation is an open parenthesis, the negation has scope over the whole WFF until the closing parenthesis.

The same holds for the scope of a quantifier phrase. A quantifier phrase has scope over the WFF immediately following it. Sometimes that WFF is atomic; sometimes, if the WFF is enclosed in parentheses, the WFF is Boolean, and the scope of the quantifier is over the whole Boolean WFF. Sometimes if the WFF is itself quantified, as in ∀x∃y(Pirate(y) ∨ SailsTo(x, tortuga)), the scope of the outer quantifier phrase ∀x is over the subsequent quantified phrase ∃y(Pirate(x) ∨ SailsTo(x, tortuga)). It works just like the scope of negation.

Now, any given WFF might have some variables in it. If the WFF is inside the scope of a quantifier phrase of a quantifier phrase that also has that variable, the variable is said to be bound by that quantifier phrase. Don't worry about the *meaning* of these WFFs yet, just consider the relationships between variables in WFFs and quantifier phrases. Let's look at some bondage now (and you thought logic was all vanilla!)

∀xPirate(x)
The x in Pirate(x) is bound by the universal quantifier phrase ∀x.

∃y Pirate(x)
The x in Pirate y is NOT bound by ∃y; in fact it is unbound.

∀aPirate(a)
There are no variables to be bound (and none to do the binding).

∀xPirate(a)

There are no variables to be bound (so the quantifier phrase is superfluous). This is "vacuous quantification". The quantifier phrase is doing no work.

$\forall x(\text{Pirate}(x) \land \text{Sexy}(x))$

Both xs in (Pirate(x) a Sexy(x)) are bound by the same quantifier phrase.

$\exists x\text{Pirate}(x) \land \text{Sexy}(x)$

The x in Pirate(x) is bound by the existential quantifier phrase Ex, but the x in Sexy(x) is unbound.

$\exists x\text{Pirate}(x) \land \exists x\text{Sexy}(x)$

Both xs are bound, but not by the same quantifier! This is important. It's as if I said 'He hit the ball. He threw it.' There is no guarantee it is the same 'he'!

$\exists x\forall y\ (\text{Pirate}\ (x) \land \text{Sexy}\ (y))$

The x and y in (Pirate (x) $\land$ Sexy (y)) are both bound.

One Caveat:

You may never bind an instance of a variable to multiple quantifier phrases. This rule could be implemented in many ways, but we will implement it at the syntactic level[2]. Nothing like $\forall x\exists x\text{Pirate}(x)$ is a WFF. That's because it would be contradictory. Is it everything that's a pirate? Or something? I call this cross-binding. Just keep in mind that logic, with all its bondage, may be kinky, but it's not that kinky. Relations between quantifier phrases and the WFFs they bind is strictly monogamous.

The easiest way to avoid this is to use a brand new variable every time you add a new quantifier phrase. Sometimes this is not necessary, but it is never wrong.

And now we can wrap this up. A variable that is unbound, that is, not within the scope of a quantifier phrase using that very variable, is free. But freedom is a bad thing! A WFF with a free variable is like a sentence with a hole in it. To say 'Pirate(x)' is like saying '___ is a pirate'. It does not attribute piracy to anything. It can't be true or false. It's not meaningful on its own.

On the other hand, a *Sentence* is a WFF with no free variables. Every individual term in a sentence is either an individual constant, or a bound variable. Sentences are the units of meaning in QL, and the bearers of truth or falsity. Instead of talking about sentences like $\forall\alpha\varphi$ or $\exists\alpha\varphi$, let's use $\forall\alpha\varphi\alpha$ and

2 I say we could implement this in different ways, because we could always allow these to be WFFs or even sentences at the syntactic level, and then at the semantic level, refuse to assign them a truth value.

$\exists\alpha\varphi\alpha$, to indcate that α is bound by the quantifier phrase.

Well, great, Smoove. You've shown us how to combine a bunch of Mr. Smartypants symbols. Big deal! What does it mean? How can I provinate?

Patience, grasshopper. The next chapter discusses meaning and then we'll be off to the races provinating.

Appendix: Quantifiers, Predicates and Higher-Order Logic

Recall that we said variables in QL only range over individuals. The do not range over the properties expressed by predicates or relations. This limits out expressive power, somewhat: How can we say "Some properties of Jeeves are properties of Wooster"? Or, consider Hume's analysis of causation: an event c causes an event e *iff*, among other things, c's and e's are constantly conjoined (c's are always followed by e's). That claim is the heart of the analysis. But c and e appear to be *names* of single events, so how do we say they are constantly conjoined? An event only happens once, so if c is followed by e, then it's trivially true that they are constantly conjoined. What we want is something like 'Events like c are always followed by events like e'. But how do you say that? 'All the events that share all relevant properties with c, are followed by events that share all relevant properties with e'. It looks an awful lot like you have to quantify over the properties of c and e to even state this, that is, you have to mention not particular properties, but you have to have variables that take those properties as values.

Logics that quantify over properties are called higher-order logics (QL is what is called first-order logic, because it only quantifies over individuals, and individuals are the first order). There is a long tradition of treating individuals – Albert, Wooster, 1 PM February 28 2016, the great fire of London – as the 'ground floor' of reality, the so-called 'first order'. Properties of individuals are associated with sets of individuals (i.e, the individuals that intuitively have the property.) So anything quantifying over that is a higher order logic.

Well, Why not quantify over that? Mostly (in my view) because of philosophical prejudice. While it's true that higher-order logics generate some nasty paradoxes, those are solvable. The real culprit is the empiricists, particularly WVO Quine. Quine just hated properties, thought they didn't exist, because they were too non-empirical. He even came up with a slogan: 'To be is to be the value of a bound variable'. That is, your theory only commits you to the sort of entities that can be quantified over, or the entities that can have variables stand in for them. So naturally Quine would not want to quantify over the values of predicates.

For further reading on this topic, see:

Rules and Definitions

Basic WFF formation rule for QL:
If φ is a relation term with an arity of n, and $\alpha_1, \ldots \alpha_n$ are exactly n individual terms, then $\varphi(\alpha_1, \ldots \alpha_n)$ is a WFF

Complex Boolean WFF Formation Rule
If φ and ψ are wffs (*atomic or otherwise!*), then

(φ ∧ ψ)

(φ ∨ ψ)

(φ → ψ)

(φ ↔ ψ)

and ¬φ (or ¬ψ)

are WFFs

Quantified Atomic WFF Formation Rule
If φ is an atomic WFF, then
∀αφ is a WFF

(where a is a metalanguage variable ranging over individual terms such as 'albert', 'wooster', 'x', 'y', and 'z'.)

Quantified General WFF Formation Rule
If φ is a WFF of any sort, then
∀αφ is a WFF

(where a is a metalanguage variable ranging over individual terms such as 'albert', 'wooster', 'x', 'y', and 'z'.)

One Caveat:

You may never bind an instance of a variable to multiple quantifier phrases. Nothing like ∀x∃xPirate(x) is a WFF.

The easiest way to avoid this is to use a brand new variable every time you add a new quantifier phrase. Sometimes this is not necessary, but it is never wrong.

A Sentence is a WFF with no free variables. Every individual term in a sentence is either an individual constant, or a bound variable. Sentences are the units of meaning in QL, and the bearers of truth or falsity.

Chapter II

What does it all mean? The Semantics of QL

There's really a very simple insight into what quantified sentences mean and what their truth conditions are. We'll lead off with that, and then explain the inner workings.

We wanted to capture the meanings of “Some” and “All”. So let's now say what our quantifiers mean: $\forall$ means all, and $\exists$ means some (technically, 'at least one'). More to the point, the corresponding quantifier phrases $\forall x$ and $\exists x$ mean, respectively, Every x ('everything') and Some x ('something' or 'at least one thing'). Even here notice how the variables are working like pronouns: 'thing' is sort of a pronoun that has the very broadest interpretation, that is, it ranges over every individual in the universe. But remember that those quantifier phrases are only meaningful in the context of some larger formula. As we did in PL, let's use Greek letters as metalanguage variables ranging over formulas. So any meaningful formula with a quantifier phrase will ultimately look like $\exists\alpha\varphi$ or $\forall\alpha\varphi$ (or it will have such a formula somewhere inside φ). This means that all we really have to do is provide truth conditions for two sorts of formulae, $\exists\alpha\varphi$ and $\forall\alpha\varphi$.

The embedded formula φ might be of three sorts: either it has no variables, it's got free variables, or it has only variables bound by the quantifier phrase. If it's got no variables, then we have a case of vacuous quantification, in which case the truth conditions are just the same as the truth conditions for $\varphi\alpha$. If it's got free variables, then either it's not a sentence, or it's part of a larger sentence that binds the variables. We'll discuss these cases later on in Chapter 7. That leaves the case of φ containing only bound variables.

Suppose φ contains bound variables. For example, φ is

Pirate(x) or
(Dropped(x) → Breaks(x))

Such a formula is, as we said, 'a sentence with a hole in it'. Formally we call this an *open sentence*. It can be read as:

___ is a Pirate
If ___ is dropped, ___(it) breaks. (___ is fragile)

Think of these sorts of formulae as representing *conditions:* the condition of being a pirate, or the condition of breaking when you're dropped (the condition of being fragile). Of course, a condition can't be true or false. But now go back to the quantifier phrase: Once we add a quantifier that binds the variable in φ, we are saying 'Everything meets the condition represented by φ' or 'Something meets the condition represented by φ'. *That* is meaningful and might be true or false! And what would make it true? Clearly, if something in the world was a pirate, or if everything in the world was fragile.

If φ is just a predicate, it's easy to see how it's a condition. It's the condition of being a squirrel, or red, or whatever. If it's a relation, it's a bit trickier – but it's the condition of being to the right of Albert, or the condition of pillaging Wooster, and so on. What about if φ is a Boolean formula, like P(x) → Q(x)?

There's only five Boolean connectives, so we only have to deal with five cases. First, conjunctions. That's easy – there are lots of conjunctive conditions, like being hard and fragile. That's such a common condition that we have a single word for it – 'brittle'. Then there are disjunctive conditions. These are sort of weird, but not that uncommon. Being made of jadite or nephrite is being jade. Being overly angry or being overly sad is being emotional. The most interesting cases are conditional conditions – these are the stuff of science! Almost all the interesting properties that science investigates are conditionals. If you put it x in a certain magnetic field, it veers to the right might be having a negative charge. If you run a current through x, it makes it to the other side of x quickly, means x is conductive. These kinds of properties are commonly called *dispositions.* Negative conditions are equally simple. The lack of water is dryness; the condition of being not sexy is being Bieberish. The reader is left to treat biconditional conditions in a similar fashion.

Oh, did we mention that the truth conditions for the Booleans are just as they were? So really there are only two new rules, and they are extremely intuitive:

A sentence of the form $\forall\alpha\varphi$ is true *iff* everything satisfies φ (i.e., *iff* φ is true of everything) A sentence of the form $\exists\alpha\varphi$ is true *iff* at least one thing satisfies j (i.e., *iff* φ is true of everything)

There is an alternative way of conceiving of this. φ is not really a sentence, but a whole range of possible sentences. For example, 'Pirate(x)' really could be:

Pirate (albert)
Pirate (wooster)
Pirate (jeeves)
etc.

Pirate(x) sort of encompasses all those sentences. Call each one a 'substitution instance' of φ. Intuitively, 'something is a pirate' or $\exists$xPirate(x) is going to be true just in case at least one of those sentences, or substitution-instances, is true (and of course you can simply use the truth rules from Volume 1 to determine this). And 'Everything is a pirate' is going to be true just in case all of those sentences are true.

While this is a useful heuristic, it is philosophically problematic. What if your language doesn't name everything? Maybe pirates are hated so much that their names are stricken from the language. That may seem like a silly example, but no language names everything. There's no name for my martini glass, for example, although there is a description of it. Most stars and planets don't have names.

A sentence of the form ∃αφ is true *iff* at least one thing satisfies j (i.e., *iff* φ is true of everything)

stars have no name. Back to our pirate example, if all the names of pirates are stricken, then there aren't any sentences like 'Pirate(blackbeard)'. But then there are no true sentences like 'Pirate(blackbeard'). So then ∃xPirate(x) is false, which is clearly ostrich logic. You can't just deny the existence of pirates by refusing to name any of them. They will pillage you anyway. So, as I said, this interpretation of the quantifiers is sometimes a useful heuristic, but don't rest too much on it.

Truth in a Model

Just as in PL, we don't want to simply tell whether or not a sentence is true (and again, of course logic alone cannot do that). We want to give the truth conditions for sentences, that is, we want to describe the relationship between the sentence and the world that holds *iff* the sentence is true. And we'd like to do it mechanically, like we did with truth tables.

But we can't quite use truth tables. Remember that a row on a truth table represents a way the world might be with respect to its atomic parts. From that we can work out whether or not a sentence is true in that world, whether it's a complex or simple sentence. And there's only a finite number of possible ways the world could turn out with respect to those atomic parts. But now things have changed.

We want to ask about sentences like 'Everything is a squirrel' (oh, joyous day!) So we basically want to run through all the things in the world and check whether they are squirrels. But how many things are there? We don't know! And there might be an infinite number of things (especially if one is talking about the mathematical world, with infinite numbers). So we really can't have any finite process like a truth table that will help us. What if we have counted a billion squirrels? Maybe we just happen to (happily) be in a squirrel-rich part of the world, and the next thing we check is actually a record executive (ugh!) We might be able to show that the sentence is false, but how could we show it is true?

So, instead of talking about whether sentences are true, we will talk about whether they are true in a world, or as we call it, true in a model. Models will roughly correspond to rows on truth tables, with the difference that there will be infinitely many relevant models. Then, we'll re-cast all the logical concepts – validity, consistency, etc., - in terms of truth in a model.

OK, so what's a model? Asking this is like asking 'What's in a world'? Well, first, there's a bunch of stuff, namely, individuals (the sorts of things that are named by individual constants, and the sorts of things that individual variables range over). We call this the *Domain* and represent it as a fancy shmancy D: $\mathscr{D}$. So every world (Model) has a bunch of stuff in it. M_1 might have a domain $\mathscr{D}$ of {albert, jeeves, wooster}. M_2 might have {jeeves, sparky, alvin}. It's a deeper philosophical question as to whether any being is in all the models – that's the question of whether anything exists necessarily. We'll assume that the answer is no, and let the metaphysicians fight it out. We'll also assume that no individuals are necessarily connected to any others. So Jeeves might be in a model without Wooster, which seems odd, but there it is.

One caveat: No models have empty domains. In other words, every world has to contain at least one thing. After all, if it didn't contain anything, how would it be any different from a non-world? This also prevents some really odd and implausible results.

In describing or modeling a world, though, we don't just want to say what things are in the world, but we want to talk about the ways they are in the world – i.e., the properties they have, the predicates that apply to them. 'But Smoove!' you say, 'That's a lot to talk about!' Well, true, but remember, just like with truth tables, we don't need to describe the whole world, just the part of it relevant to the sentences we're evaluating. To do this, all we need to do is list off all the predicates that occur in the sentences at hand, and then give their extension – the list of things that have them. This is done via an Interpretation, $\mathscr{I}$, which is simply a function that assigns random individuals to each predicate. In essence, $\mathscr{I}$ tells you what things are squirrels in the model, what things are pirates, what things are tall, what things pillage what other things, and so on. In the case of relation terms, $\mathscr{I}$ provides not a list of individuals, but a list of pairs of individuals – the pairs where the first individual stands in R to the second individual[3]. For example, $\mathscr{I}$ (Pillages) might be {<jeeves, wooster>,<albert, hanna>}, indicating that in this model, jeeves pillages wooster, and albert pillages hanna. For example:

$\mathscr{M}1 = \langle \mathscr{D}, \mathscr{I} \rangle$
$\mathscr{D}$ = {jeeves, wooster, sparky}
$\mathscr{I}$(Squirrel) = {sparky}
$\mathscr{I}$(Pirate) = {jeeves, wooster}
$\mathscr{I}$(Pillages) = {<sparky, wooster>, <jeeves, sparky>}

This describes a world or model with three individuals – Jeeves, Wooster, and Sparky. Sparky (and only sparky) is a squirrel, Jeeves and Wooster are Pirates. Sparky pillages Wooster, and Jeeves in turn pillages Sparky. For all the fanciness, a model is just a shorthand way to specify a world.

For ordinary, non quantified sentences, the truth conditions are exactly what you would expect. An atomic sentence $\varphi(\alpha_1,...\alpha_n)$ is true in a model $\mathscr{M}$ just in case the objects named by $\alpha_1,...\alpha_n$ are in the extension of φ (as given by $\mathscr{I}\varphi$). Which is just to say that the sentence is true in $\mathscr{M}$ *iff* the relevant objects really have the properties predicated of them in the world $\mathscr{M}$. And guess what? Nothing changes with the Boolean connectives – their truth conditions are exactly the same as in PL.

Let's work through some examples. First, let's take a simple atomic sentence, and provide a model in

3 Or, for 3-place relations, a set of triples, for 4-place relations, a set of quadruples, and so on. Generally, we can say that $\mathscr{I}$ assigns a relation term R of arity n, a set of ordered n-tuples.

which it is true, and a model in which it is false. How about,

Between (tortuga, port royale, kingston).

A model where it is true:

$\mathscr{M}1 = \{\mathscr{D}1, \mathscr{I}1\}$
$\mathscr{D}1$ = {tortuga, port royale, kingston, nassau} (you have to have all the individuals mnetioned in the sentence being evaluated, I threw in nassau for kicks)
$\mathscr{I}1$(Between) = {< tortuga, port royale, kingston>, <kingston, port royale, nassau>}

In our model $\mathscr{M}1$, is the ordered triple < tortuga, port royale, kingston> in the extension of Between? It is easy to check – just look at $\mathscr{I}1$(Between). Sure enough, < tortuga, port royale, kingston> is the first element in that extension. So yes, Between (tortuga, port royale, kingston) is true in $\mathscr{M}1$.

Now a model where Between (tortuga, port royale, kingston) is false:

$\mathscr{M}2 = \{\mathscr{D}2, \mathscr{I}2\}$
$\mathscr{D}2$ = {tortuga, port royale, kingston, nassau}(same reasoning as with M1)
$\mathscr{I}2$(Between) = {< tortuga, kingston, port royale>, <kingston, port royale, nassau>}

In our model, M2, is the ordered triple < tortuga, port royale, kingston> in the extension of Between? Again, it is easy to check – just look at $\mathscr{I}2$(Between). < tortuga, port royale, kingston> is not there! <tortuga, port royale, kingston> is there, but order matters! Saying Jeeves pillages Wooster is very different from saying Wooster pillages Jeeves.

Now let's do something slightly more complicated, with a connective: If Albert is a squirrel, Albert is sexy.

Squirrel(albert) $\rightarrow$ Sexy(albert)

First, a model where this is true:

$\mathscr{M}3 = (\mathscr{D}3, \mathscr{I}3)$
$\mathscr{D}3$ = {albert, sparky, hanna}
$\mathscr{I}3$(Squirrel) = {sparky}
$\mathscr{I}3$(Sexy) = {hanna}

In $\mathscr{M}3$, is Squirrel(albert) true? No – the only squirrel is Sparky. Is Sexy(albert) true)? No – Hanna is the only sexy thing in $\mathscr{M}3$. But remember the truth conditions for → : if the antecedent is false, the whole conditional is true. And the antecedent is false in $\mathscr{M}3$, so Squirrel(albert) → Sexy(albert) is true in $\mathscr{M}3$.

Now let's consider a world where it's false. We'll be lazy, and start by keeping as much of $\mathscr{M}3$ as possible.

$\mathscr{M}4 = (\mathscr{D}4, \mathscr{I}4)$
$\mathscr{D}4$ = {albert, sparky, hanna}
Let's remind ourselves of the conditions where a conditional is false. It's only false when the antecedent is true and the consequent is false. So in our world we want Albert to be a squirrel, and not sexy.
$\mathscr{I}4$(Squirrel) = {albert, sparky}
$\mathscr{I}4$(Sexy) = {hanna}

Indeed, in $\mathscr{M}4$, Albert is in the extension of squirrel, and not in the extension of sexy; Albert is a squirrel, yet not sexy. Farfetched though it may be to imagine an unsexy squirrel, perhaps in $\mathscr{M}4$ Albert has leprosy or something.

Giving a model where a sentence is false (or an argument is shown to be invalid) is called giving a countermodel. It is excellent practice for doing what philosophers and lawyers do – poking holes in arguments by finding counterexamples. The more complex the sentence / argument, the trickier it is to find countermodels. One might reasonably ask here: if one hasn't found a countermodel, is that enough to show an argument is valid? How do you know the next model will not be the countermodel? After all, when we did truth tables, we only had to consider finitely many worlds (rows on a truth table). But there are infinite many models! We'll consider this in the appendix.

The Variable Value Assignment Function

So far, we haven't *really* added anything to the semantics of PL. We've slightly expanded on the idea of truth tables, to become models. But we have nothing so far that will help us with evaluating the truth conditions of quantified statements.

Remember that our guiding intuition is that a sentence of the form $\exists\alpha\varphi\alpha$ is true *iff* everything meets condition φ, and a sentence of the form $\forall\alpha\varphi\alpha$ is true *iff* at least one thing meets condition φ. One way we can capture this is by treating the variables as being just like names, but imagining that they refer to all sorts of individuals. If the embedded sentence φα is true no matter what α refers to, then everything is φ. If the embedded sentence is true for one possible thing that α could refer to, then something is φ. So, we introduce the idea of a 'variable value assignment function' $\mathscr{V}_n$. This is simply a function that takes the variables in a formula, and randomly assigns them to refer to individuals in the domain. For example,

$\mathscr{M}1 = \{\mathscr{D}1, \mathscr{I}1\}$
$\mathscr{D}1$ = {sparky, rodney, trump}
$\mathscr{V}_1$x = trump
$\mathscr{V}_2$x= sparky
$\mathscr{V}_3$x = rodney
(I leave out $\mathscr{I}_1$ here, it's not relevant to this part)

This may look intimidating but it's really just a fancy-pants way of talking about all the different things a variable could refer to. Now we can formalize our earlier thought that

A sentence of the form $\exists\alpha\varphi\alpha$ is true *iff* everything meets condition φ, and

A sentence of the form $\forall\alpha\varphi\alpha$ is true *iff* at least one thing meets condition φ.

The idea again is that $\exists\alpha\varphi\alpha$ is true *iff* $\varphi\alpha$ is true on at least one way of considering what the variable α refers to. And $\forall\alpha\varphi\alpha$ is true *iff* $\varphi\alpha$ is true on every way of considering what the variable α refers to. Or, in English, 'Something is φ [a pirate]' is true just in case 'x is a pirate' is true for at least one thing x. Formally:

A sentence of the form $\exists\alpha\varphi\alpha$ is true in $\mathscr{M}$ *iff* $\varphi\alpha$ is true for at least one $\mathscr{V}n(\alpha)$ in $\mathscr{M}$
A sentence of the form $\forall\alpha\varphi\alpha$ is true in $\mathscr{M}$ *iff* $\varphi\alpha$ is true for every $\mathscr{V}n(\alpha)$ in $\mathscr{M}$

Let's again run through some examples. We'll just take the examples from our previous, non-quantified sentences, and turn them into quantified sentences. (We call this process *quantifying over* a particular individual constant).

$\exists$xBetween (x, port royale, kingston).

A model where it is true:

$\mathscr{M}1 = \{\mathscr{D}1, \mathscr{I}1\}$
$\mathscr{D}1$ = {tortuga, port royale, kingston, nassau} (you have to have all the individuals mnetioned in the sentence being evaluated, I threw in nassau for kicks)
$\mathscr{I}1$(Between) = {< tortuga, port royale, kingston>, <kingston, port royale, nassau>}
$\mathscr{V}_1$(x) = tortuga
$\mathscr{V}_2$(x) = port royale
$\mathscr{V}_3$(x) = kingston
$\mathscr{V}_4$(x) = nassau
(In practice, we never really list off the Vns. It's tedious, and they are obvious)

This is a sentence of the form $\exists\alpha\varphi\alpha$, so it's true *iff* $\varphi\alpha$ is true under at least one $\mathscr{V}$. Is it? Look at $\mathscr{V}_1$. That assigns x the value of Tortuga, so under $\mathscr{V}_1$, <x, port royale, kingston> is just <tortuga, port royale, kingston>. Is that in the extension of $\mathscr{I}1$(Between)? Why, yes it is! Go figure. So, yeah, on at least one reading of x, Between(x, port royale, kingston) is true - in - $\mathscr{M}1$. Which is just to say, at least one thing is between Port Royale and Kingston.

Now let's try a countermodel. Again we'll be lazy and stick as close as we can to the original model $\mathscr{M}1$:

$\mathscr{M}2 = \{\mathscr{D}2, \mathscr{I}2, \mathscr{V}\}$
$\mathscr{D}2$ = {tortuga, port royale, kingston, nassau}
$\mathscr{I}2$(Between) = {< tortuga, port royale, kingston>, <kingston, port royale, nassau>}
$\mathscr{V}1$(x) = tortuga
$\mathscr{V}2$(x) = port royale
$\mathscr{V}3$(x) = kingston
$\mathscr{V}4$(x) = nassau

If ∃xBetween (x, port royale, kingston) is false, then nothing is between Port Royale and Kingston. So if it's false no value of x can make a triple that is in the extension of Between.

What are the possible assignments of (x, port royale, kingston)? There's only four in $\mathscr{M}_2$:

(tortuga, port royale, kingston)
(port royale, port royale, kingston)
(kingston, port royale, kingston)
(nassau, port royale, kingston)

That's just based on the possible readings of x. Are any of those four in the extension of $\mathscr{I}_2$(Between)? It's simple enough to check: No.

Finally, let's try a quantified Boolean sentence: All squirrels are pirates.

$\forall x(S(x) \rightarrow P(x)$

Here's a model where it's true:

$\mathscr{M}1 = \{\mathscr{D}1, \mathscr{I}1\}$

$\mathscr{D}$1= {albert, sparky, trump}
$\mathscr{I}$1(Squirrel) = {albert, trump}
$\mathscr{I}$1(Pirate) = {albert, sparky, trump}

The sentence is of the form $\forall x\varphi(x)$, so it's true in $\mathscr{M}$1 *iff* $\varphi(x)$ is true in $\mathscr{M}$1 under every variable assignment $\mathscr{V}$. This is why it pays to start by using models with small domains – there are fewer variable assignments to check. In this case, there's only 3 – one assigning Albert to x, one assigning Sparky to x, and one assigning Trump to x. As it turns out, φ is itself complex – it's a conditional. That's okay, think of it as a disposition. Is φ(x) true when we read x as referring to Albert? That would mean

Squirrel(albert) → Pirate(albert)

Is Albert in the extension of $\mathscr{I}$1(Squirrel)? Yes. Is he in the extension of $\mathscr{I}$1(Pirate)? Also yes. So Squirrel(albert) → Pirate(albert) is true in $\mathscr{M}$1. (We know this just from our old Boolean truth definitions).

Move on to the next reading of x, and suppose x refers to sparky:

Squirrel(sparky) → Pirate(sparky)

Is sparky in the extension of $\mathscr{I}$1(Squirrel)? No! So Squirrel(sparky) is false. And we know from Boolean logic that when the antecedent is false, the whole conditional is automatically true. We don't even have to check whether sparky is in the extension of $\mathscr{I}$1(Pirate). So

Squirrel(sparky) → Pirate(sparky) is true in $\mathscr{M}$1.

The only other assignment of values to x is the one that pretends x refers to Trump. So under that assignment, φ(x) is

Squirrel(trump) → Pirate(trump)

Is Trump in the extension of $\mathscr{I}1$(Squirrel)? Yes. is he in the extension of $\mathscr{I}1$(Pirate)? Also yes. So

Squirrel(trump) $\rightarrow$ Pirate(trump) is true in $\mathscr{M}1$.

(Squirrel(x) $\rightarrow$ Pirate(x)) is thus true in $\mathscr{M}1$ no matter what x refers to, or, to be smarty-pants about it, for every assignment of values to the variable x. So, by our definition of truth for universally quantified sentences,

$\forall$x(S(x) $\rightarrow$ P(x) is true in $\mathscr{M}1$.

We leave it to the reader to construct a model $\mathscr{M}2$ in which $\forall$x(Squirrel(x) $\rightarrow$ Pirate (x)) is false. Hint: all you need to do is have a model where at least one thing that is a squirrel is not a pirate.

You may have noticed we haven't discussed sentences with multiple variables or multiple quantifiers. While the principles for those are exactly the same, they are quite tricky, and we're saving them for a later chapter. Let's revisit the logical concepts for now.

Rules and Definitions

A sentence of the form $\forall\alpha\varphi$ is true *iff* everything satisfies φ (i.e., *iff* φ is true of everything)

A sentence of the form $\exists\alpha\varphi$ is true *iff* at least one thing satisfies j (i.e., *iff* φ is true of everything)

A sentence of the form $\exists\alpha\varphi\alpha$ is true in $\mathscr{M}$ *iff* $\varphi\alpha$ is true for at least one $\mathscr{V}$n(α)
A sentence of the form $\forall\alpha\varphi\alpha$ is true in $\mathscr{M}$ *iff* $\varphi\alpha$ is true for every $\mathscr{V}$n(α)

Chapter III

The Logical Concepts – Now For Quantifiers Too!

"But Smoove!" you ask, "What the heck have we *been* doing, if not logical concepts?" Yeah, fair enough. But just like with propositional logic, I'm talking here about the central concepts of logic – validity, entailment, consistency, and so on. The good news here is that they are almost entirely unchanged from propositional logic, except we use models instead of truth tables. In fact, the informal definitions are entirely unchanged (if you don't remember what those are, go back and re-read chapter II of the first volume). Recall, the concepts are: Equivalence, Validity, Entailment, Logical Truth, Contingency, and Consistency.

A) Logical Equivalence.

Recall that two sentences of PL are logically equivalent *iff* they are true on all and only the same rows of a truth table. A row on a truth table is a picture of a possible world in PL; in QL, a picture of a possible world is a model. So, unsurprisingly, two sentences of QL are logically equivalent *iff* they are true in all and only the same models:

Logical Equivalence
Two sentences φ and ψ are logically equivalent *iff* they are true in all and only the same models.

To show that two sentences are *not* equivalent, one produces a *countermodel* – a model where one sentence is true and the other is false. Simple enough. How do you prove that two sentences really *are* logically equivalent? Show that every model in which one is true, the other is. Hmmm. How do do you show that? How many possible models are there? That's right – an infinite number of possible models. Suppose you show that in the first ten models you look at where the one sentence is true, the other is also. That's already a really tedious task, ugh! But there's no guarantee that the next model you look at. . or the one after that. . .or the one after that. . .will not turn out to be a countermodel.

This highlights an important property of QL. QL has no finite decision procedure. If two sentences are equivalent, there's no finite way to show it. You can't do it with models, and we'll see that you can't do it with proofs. If they are equivalent, you can prove it with proofs, but it won't be an automatic process like it was in PL. It's possible to get stuck in infinite loops in a proof and never really get where you want to go. Because of this, QL is much more of an art than PL. PL can be done on autopilot – QL often requires some cleverness and intuition.[4]

Let's try an example. Consider the two sentences, 'Everything is either a squirrel or a pirate', and 'Either everything is a squirrel or everything is a pirate'. I realize we haven't done translations yet, but I hope a bit of thought convinces you that these two sentences are

∀x(Squirrel(x) ∨ Pirate(x)) and
∀xSquirrel(x) ∨ ∀xPirate(x)

respectively. What's the difference? The former seems to allow for a world with a mixture of squirrels and pirates, as long as everything is one or the other. The latter is more exclusionary: It allows for a world entirely composed of squirrels, or entirely composed of pirates, but not a mixture (except insofar as the mixture involves only pirate squirrels).

So, model time. Consider $\mathscr{M}$1:

$\mathscr{M}$1 = {$\mathscr{D}$1, $\mathscr{I}$1}
$\mathscr{D}$1 = {albert, sparky} (Again, we'll start with a small domain just for the sake of ease)
$\mathscr{I}$1(Squirrel) = {albert}
$\mathscr{I}$1(Pirate) = {sparky}

4 Actually, the situation is not quite so dire. If the sentences only involve one-place predicates, you can find countermodels to arguments (if they exist) within a finite number of models. So if you run through all those models and find no countermodels, the argument is valid. The problem is that the number of models starts to get really enormous as you add atomic sentences to the argument. So it's a finite process, but it might be so many models that it's not really doable in practice.
On the other hand, when you add in relations, all bets are off.

Does everything in the domain satisfy the condition of being a squirrel or a pirate, i.e.,

Pirate(x) ∨ Squirrel(x)?

That is, is Pirate(x) ∨ Squirrel(x) true on every assignment of values to x? There are only two such assignments, because Sparky and Albert are the only things in $\mathscr{M}$1. So, is

Pirate(x) ∨ Squirrel(x)

true if x refers to Albert? Yes, Albert is a squirrel in $\mathscr{M}$1, so Albert is either a pirate or a squirrel in $\mathscr{M}$1 (by the definition of truth for 'or'). Similarly, Sparky is a pirate in $\mathscr{M}$1, so Sparky is either a squirrel or a pirate in $\mathscr{M}$1, i.e.,

Pirate(x) ∨ Squirrel(x)

is true in $\mathscr{M}$1 if x refers to Sparky. There is nothing else for x to refer to in M1, so

Pirate(x) ∨ Squirrel(x)

is true in $\mathscr{M}$1 no matter what x refers to, i.e., it is true under every assignment of values to variables. But that means that

∀x(Squirrel(x) ∨ Pirate(x))

is true in $\mathscr{M}$1 (by the definition of truth for universal quantifiers).

So far, so good. Let's look at that second sentence,

∀xSquirrel(x) ∨ ∀xPirate(x)

Is that true in $\mathscr{M}$1? Let's think a second. What is the central connective, or the connective with the widest scope? It's a trick! The disjunction has the widest scope – it disjoins two universals! So first we look at each universal. If either universal is true, then the disjunction is true (by the definition of truth for disjunctions). If neither is true, the disjunction is false.

First, ∀xSquirrel(x). As before, there are only two possible values of x – namely, Sparky and Albert. Is Albert a squirrel in 𝓜1? A quick look at 𝓘1 tells us yes. Is Sparky a squirrel in 𝓜1? No – an equally quick look at 𝓘1 tells us that Sparky is not a squirrel. So

Squirrel(x)

is *not* true for every value of x. Thus,

∀xSquirrel(x)

is not true in 𝓜1. All is not over, though – only one half of the disjunction needs to be true for the whole disjunction to be true. Perhaps everything is a pirate in 𝓜1? No – Albert is not a pirate in 𝓜1. So there is at least one way of taking Pirate(x) that makes it false in M1; i.e., there's one assignment of values to x under which Pirate(x) is not true. So, not every way of reading Pirate(x) in 𝓜1 is true, so,

∀xSquirrel(x)

is false in 𝓜1. So both

∀xPirate(x) and
∀xSquirrel(x)

are false in 𝓜1; by the definition of truth for disjunctions,

∀xSquirrel(x) ∨ ∀xPirate(x)

is false in 𝓜1. So there's a model where

∀xSquirrel(x) ∨ ∀xPirate(x) = F and
∀x(Squirrel(x) ∨ Pirate(x)) = T.

Thus, the two sentences are not logically equivalent – they are not merely two ways of saying the same thing, because it's possible for one to be true and the other false. But this bears out our informal reasoning about whether or not they really say the same thing. Notice that you only need one stinking countermodel to show they aren't equivalent, just like you only needed on line on a truth table.

What if they'd both been true in $\mathscr{M}1$? That would show nothing. After all, M2 might turn out to be a countermodel. If they are equivalent, they "march in lockstep", they always share the same truth value, because they mean the same thing. To show that two sentences *are* logically equivalent, we will need to wait for QL proofs.

B) Validity

Remember that validity is a property of arguments. Arguments are valid (informally) when the truth of the premises guarantees the truth of the conclusion; i.e., if there's no way for the premises to be true without the conclusion being true. This lends itself very nicely to a treatment using models. Formally,

Validity
An argument is valid *iff* every model in which the premises are true, the conclusion is also true.

This has the same limitations as the method for checking logical equivalence – if an argument is invalid, you can prove it with a countermodel, but if it's valid, you're out of luck and have to make a formal proof.

Let's try an argument. All pirates are sexy, Albert is sexy, so Albert is a pirate. Again, we haven't done translations, but this is simple enough:

∀x(Pirate(x) → Sexy(x))
Sexy(albert)
∴ Pirate(albert)

Who thinks this is valid? Well, we'll see.

Ordinarily, you'd probably try a few models here until you started to see what it was going to take to make a countermodel. I'll cut straight to the chase, though.

$\mathscr{M}$1 ={$\mathscr{D}$1, $\mathscr{I}$1}
$\mathscr{D}$1 = {albert, jack}
$\mathscr{I}$1(Pirate) = {jack}
$\mathscr{I}$1(Sexy) = {albert, jack}

We know we have to put Albert in the extension of sexy. Why? Because if $\mathscr{M}$1 is a countermodel, the premises must be true and the conclusion false. And Sexy(albert) is one of the premises, so Albert better be sexy in $\mathscr{M}$1. We also know that everything that's a pirate is sexy, so everything in the extension of pirate had also better be in the extension of sexy. A quick check shows that this is so – on every assignment of values to x, if that value is in the extension of pirate, it's also in the extension of sexy. Specifically, the only pirate, Jack, is also sexy. (A cheap way to achieve this is just to have no pirates!) So in $\mathscr{M}$1, both premises are true. What about the conclusion, Pirate(albert)? No – and no quantifiers to mess with, just look at $\mathscr{I}$1(Pirate), and see that albert ain't in the extension. So in $\mathscr{M}$1, the premises are true but the conclusion false – the argument is invalid. Just like in PL, you only need one countermodel to show this.

Pro tip: if you have simple, unquantified sentences in your premises, use those to start building your model.

C) Entailment

The premises entail the conclusion *iff* the argument is valid. So we just use the method of model/ countermodel for validity to demonstrate entailment. If there's a model where the premises are true and the conclusion false, the premises do not entail the conclusion.

Entailment
premises entail a conclusion *iff* in every model where the premises are true, the conclusion is also true.

D) Logical Truth / Logical Falsity

Remember, in PL, a sentence is logically true (false) *iff* it's true (false) in every possible circumstance. And in PL a circumstance is modeled as a line on a truth table. Since QL models these possible circumstances with models, in QL,

Logical Truth (Falsity)
A sentence is logically true (false) if it is true (false) in every possible model.

Again, this method has the same limitations as the model method for Validity and Equivalence. Bummer. Actually, that's not so bad – just like with truth tables, you will get sick of models and how tedious they are. Let's see. How about,

$\forall x Pirate(x) \leftrightarrow \exists x \neg Pirate(x)$

Everything is a Pirate, if and only if something is a non-pirate. That's certainly not logically true, although it may be logically false (hint: it is logically false). Back to the runway:

$\mathscr{M}1 = \{\mathscr{D}1, \mathscr{I}1\}$
$\mathscr{D}1 = \{sparky, wooster\}$
$\mathscr{I}1(Pirate) = \{wooster\}$

Hey, that's a pretty small model. Maybe this will be easier than we thought!

Is everything a pirate? That is, does every assignment of values to 'x' make Pirate(x) turn out true?[5]

5 At this point, I am going to stop using the formalistic language of value assignments and so forth, and revert to more

No, Sparky is not a pirate. So the left-hand side of the biconditional is false. What about the right-hand side, ∃x¬Pirate(x)? This says something is a non-pirate (NOT: nothing is a pirate!) And in fact something, namely Sparky, is a non-pirate. So ∃x¬Pirate(x) is true in $\mathscr{M}1$. But then it is simple enough to evaluate the biconditional – biconditionals are only true when each side has the same truth value. But the left side is false and the right side is true in $\mathscr{M}1$. So

∀xPirate(x) ↔ ∃x¬Pirate(x)

is false in $\mathscr{M}1$. If it's false in $\mathscr{M}1$, it's not true in every model, so it's not logically true.

What if we made a slight modification, to, say,

∀xPirate(x) ↔ ¬∃x¬Pirate(x)?

It's easy enough to see that this is true in $\mathscr{M}1$. In fact, it's also true in $\mathscr{M}2$, and $\mathscr{M}3$. "But Smoove," you ask, "how do you know? You haven't even given $\mathscr{M}2$ and $\mathscr{M}3$ yet!" True Dat. But I also know that it's a logical truth for independent reasons (Hint: if everything is a pirate, then there cannot be any non-pirates!) So I know that it will be a fool's errand to try to find a countermodel. Still, does the fact that I can't find a countermodel prove anything? No. Maybe I just haven't had enough coffee. Maybe the next model I look at will be the countermodel. (It won't, but I can't prove that using models).

Conversely, to show that a sentence is not logically false, all one has to do is produce a single model where that sentence is true. We leave this exercise to the student.

E) Contingency

Recall that a contingent sentence is a sentence whose truth is contingent on the way the world turns out – it could be true, and it could be false. In PL, that was a sentence that was true on at least one line

informal language like 'Is everything a pirate?' If I revert to formalism, it's because I wish to belabor a point (or make a new one).

of a truth table, and false on one line. So I sincerely hope it comes as no surprise to learn that in QL, a sentence is contingent *iff* it's true in at least one model, and false in at least one other model.

Contingency
A sentence is contingent *iff* it is true in at least one model, and false in at least one other model.

There's not really any new technique here, so it doesn't seem worthwhile to work through an example.

F) Consistency

Recall that our informal definition of consistency was that the consistent sentences don't rule each other out. That is, it was *possible* for consistent sentences to all be true together (even if they aren't all *actually* true). In PL, that meant that there was at least one row on the truth table where all the sentences in question were true. Surprise! In QL sentences are consistent *iff* there's at least one model in which they are all true:

Consistency
A set of sentences Γ is consistent *iff* there is at least one model in which every sentence φ ε Γ is true.

While there is no real new technique here, it seems like people get hung up the most on consistency, so let's work through an example

All Republicans drink gin, everything is a Republican or a Green, Rodney drinks gin.

$\forall x(R(x) \rightarrow DG(x))$
$\forall y(R(y) \vee G(y))$
$DG(rodney)$

Each of these sentences is *actually* false[6] (unless my cat has been sneaking around and hitting my gin

6 i.e., false in the model that best represents the actual world.

supply unbeknownst to me). But could they be all true together? A little reflection convinces me that they don't rule each other out, but other sets of sentences aren't so easy to evaluate. So if they don't rule each other out, they are consistent with each other, and there is a model where they are all true.

$\mathscr{M}1 = \{\mathscr{D}1, \mathscr{I}1\}$
$\mathscr{D}1$ = {rodney, sparky, trump}
$\mathscr{I}1$(DG) = {rodney, trump}
$\mathscr{I}1$(R) = {trump}
$\mathscr{I}1$(G} = {rodney, sparky}

DG(rodney) is easy. Rodney is in the extension of $\mathscr{I}1$(DG), as can be easily checked. How about

$\forall y(R(y) \vee G(y))$?

Well, there's only three things in the domain – Sparky, Rodney and Trump. Trump is a republican in $\mathscr{M}1$, and Sparky and Rodney are greens. So, in $\mathscr{M}1$, everything is one or the other of a Republican or a Green, and so $\forall y(R(y) \vee G(y))$ is true in $\mathscr{M}1$. How about that first sentence,

$\forall x(R(x) \rightarrow DG(x))$?

Well, as we saw, there's only one Republican (Trump). So if this sentence is true in $\mathscr{M}1$, then trump better drink gin (since the sentence says that all republicans drink gin). Sure enough, trump is in the extension of $\mathscr{I}1$(DG)[7]. This goes a long way towards explaining Trump's garish design aesthetic. As it turns out, Rodney also drinks gin in $\mathscr{M}1$. Is this a problem? No, we said that all Republicans drink gin – that doesn't mean they have a lock on the stuff. That would be the different claim that *only* Republicans drink gin (for the curious, that would be $\forall x(DG(x) \rightarrow R(x))$).

Thus, in $\mathscr{M}1$, all three sentences are true. There's at least one way for the sentences to all be true together, or at least one model in which they are all true. So they are consistent. How would you show

7 It's not really relevant, but I have been informed that Trump does not drink. And he likes his steaks cooked well done.

they are inconsistent? You'd be in the same boat as before – simply giving a model, or ten models, or a hundred models, where the sentences were *not* all true, would not be enough. Who knows? Maybe the next model is the one that does it.

Cool. You are now sick of models, so let's move on to the artistic part of QL – translation. ("Haha, Smoove, you have a funny conception of art"). We will return to models briefly when we do the logic of relations.

Rules and Definitions

Logical Equivalence
Two sentences φ and ψ are logically equivalent *iff* they are true in all and only the same models.

Validity
An argument is valid *iff* in every model where the premises are true, the conclusion is also true.

Entailment
premises entail a conclusion *iff* in every model where the premises are true, the conclusion is also true.

Logical Truth (Falsity)
A sentence is logically true (false) if it is true (false) in every possible model.

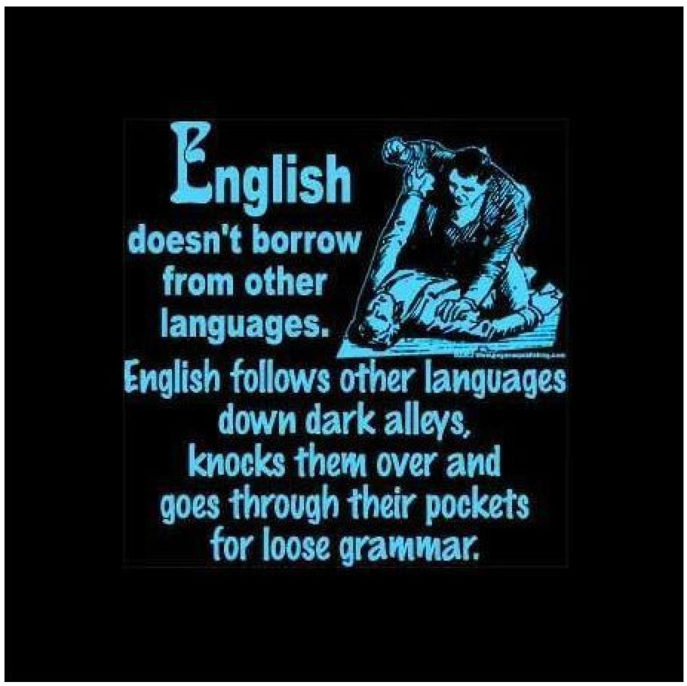
Contingency
A sentence is contingent *iff* it is true in at least one model, and false in at least one other model.

Consistency
A set of sentences Γ is consistent *iff* there is at least one model in which every sentence φ ε Γ is true.

Chapter IV

The Art of Translation

Welcome to the absolute hardest part of logic. You thought translation was easy in PL, and it was. Things go up a notch in QL. By the time we get to QL with relation terms it gets downright hairy. Is this a problem of logic? No; the problem is with the natural language (English, in this case) that we are translating from.



English is a nasty, complicated beast, and it would be fairly impossible to teach the semantics of English in a logic textbook. In the end, we have to rely on our intuitive understanding of the logic of English. But we can offer heuristics, rules of thumb, and other devices to sharpen your native understanding of both the natural and the formal languages.

Boolean sentences and quantified sentences

The first thing to consider in translation is whether you have a Boolean sentence or a quantified sentence. Not all sentences with quantifiers are quantified sentences!

Think back to the early days of propositional logic, when we talked about the central connective of a sentence (and associated sentences with their central connective). The central connective is the connective attached to the outermost parentheses of the sentence. So

$(P \rightarrow Q)$ is a conditional,
$((P \rightarrow Q) \vee (S \leftrightarrow T))$ is a disjunction[8]

and so on. There was one difference – negations. That's because the connectives connect two sentences, while negation works on a single sentence. But the principle remains – if a negation is outside the outermost parentheses, the negation is the central "connective", and the sentence is a negation. If it's inside, it's isn't the central connective. So,

$\neg ((P \rightarrow Q) \vee (S \leftrightarrow T))$

is a negation, and instead of saying that the negation is the central connective, we say it has the widest scope. But it's the same idea.

$(\neg(P \rightarrow Q) \vee (\neg S \leftrightarrow T))$

on the other hand, is not a negation! The first negation only applies to $(P \rightarrow Q)$, and the second negation only applies to S. The sentence as a whole is a disjunction of two clauses, $\neg(P \rightarrow Q)$ and $(\neg S \leftrightarrow T)$. So not all sentences with negations in them are themselves negations.

So it goes with quantifiers. Think of quantifiers as being sort of like negations, in that they don't connect sentences, they work on single sentences – whether those single sentences are complex

8 Remember that in practice we leave the outermost parentheses off, so beware of missing but implied parentheses!

sentences set off by parentheses, or not. So if a quantifier is outside the parentheses of a sentence, we have a quantified sentence. But just like negations, if the quantifier is inside the sentence and applies to only part of the sentence, it's a Boolean sentence (unless there's a further quantifier on the outside). For example:

∀xPirate(x)

is a universally quantified sentence. Alright, there's no parentheses, but Pirate(x) is (the only)single sentence and ∀x has scope over it.

∀x((Pirate(x) ∧ Ninja(albert)) → BetterThan(x, albert))

Be careful with the parentheses! ∀x has scope over the whole thing. It too is a quantified sentence. But now look at

(∀xPirate(x) → ∃yMerchant(y))

Everything is a pirate only if something is a merchant. Well, that seems true. You can't have any pirates if there are no merchants to pirate from. But everything is *everything*, so this means that that merchant (who is part of everything) must also be a pirate. OK, even pirates have to load up ships full of booty to go sell, so I guess they are merchants when they do that. Anyway our discursus on the economics of piracy digresses a bit. The point is that the sentence is a conditional – the antecedent is a universal, and the consequent is an existential, but the sentence as a who connects the two on a conditional relationship.

So, not every sentence with a quantifier is a quantified sentence.

The trick in translation lies in identifying whether your sentence is a quantified sentence or a Boolean. This will give you the “skeleton” of your sentence. You'll find that especially where only one-place predicates occur, it's pretty easy to do translations – they are fairly automatic. But translations get rapidly harder as the nastiness and complexity of the English language is being modeled. So finding the skeleton is important. What you want to do is find the skeleton, and then gradually fill in the parts of the sentence. It's helpful to use the skeleton, then use sentence letters as “placeholders” for the parts and work the subparts out.

If the skeleton is a Boolean, there are no new ideas. In fact, for any subsentence that's a Boolean, there are no new ideas. The real innovations come with quantifiers.

The Four Aristotelean Forms

Modern logicians tend to be a pretty snobby lot of nerds with think that nothing interesting happened in logic until 1879. Yeah, sure, Aristotle did laid some of the obvious groundwork, but that square of opposition for dealing with quantification was a godawful mess. Nonetheless Aristotle, who I understand was an OK philosopher, made a brilliant discovery about quantifiers. Almost all interesting quantified phrases can be put in 1 of the following 4 categories:

All Fs are Gs
No Fs are Gs
Some Fs are Gs
Some Fs are not Gs.

I won't bore you with an argument for this claim, but just see if you can come up with a complex quantified sentence that doesn't take one of those forms. Okay, that's all well and good, but what are these in QL? Let's look one by one.

All Pirates are sexy

That's got a universal quantifier in it, and it's at the start of the sentence, so it looks like its going to be a universally quantified sentence (i.e., of the form $\forall x \varphi x$). So what's φ?

This clearly says something about pirates. Does anything follow from being a pirate? Why, yes, sexiness! Well, there's that word “follow from”. Conditional, right? If you're a pirate, you're sexy! Or:

$\forall x(\text{Pirate}(x) \rightarrow \text{Sexy}(x))$

Here's another way to think about it. The universal quantifier says to consider the whole world. The antecedent, Pirate(x), says to consider just the pirate part of it. The consequent says, that part is sexy.

All Fs are Gs
$\forall x(F(x) \rightarrow G(x))$

No Pirates are Sexy

There are actually two acceptable translations of this. “But Smoove!” you howl. “I thought you said that logic was pure and unambiguous!” Fret not! We will prove that each translation is logically equivalent. Later.

First, you could think of the sentence as reading “some pirates are sexy – NOT!” and read it simply as a negation of Some Pirates are Sexy (which we will do next). Alternatively, you could read it as making a claim about all pirates. Suppose something is a pirate. What do we then know (if this sentence is true)? Right, they're unsexy. So it's a variation on All Fs are Gs:

$\forall x(\text{Pirate}(x) \rightarrow \neg\text{Sexy}(x))$

No Fs are Gs
$\forall x(F(x) \rightarrow \neg G(x))$

Some Pirates are Sexy

Well, 'some' at the beginning of the sentence is a giveaway that we have an existentially quantified sentence here. The temptation is to read this as a condition: Ex(Pirate(x) → Sexy(x)). What does that mean, though? Pick someone – if he's a pirate, he's sexy? No, that's actually a universal, because it's saying something about whoever you pick no matter who they are. Someone has this property – if they are a pirate they are sexy? Whoa, that's way too easy. My cat Sparky had that property. Huh? Well, Sparky isn't a pirate, so (Pirate(sparky) → Sexy(sparky)) is automatically true! (False antecedent, true conditional remember!) If it's true for Sparky, it's true for at least one value of x, so the existential would be true. And that holds *even if there are no sexy pirates in the universe!* So if the QL sentence is true even in models where there are no sexy pirates, it can't be a good translation.

So back up. We want it to be true in models where there's at least one sexy pirate. That is, one thing has the property of being sexy and also being a pirate. That ought to give it away:

Ex(Pirate(x) ∧ Sexy(x))

Let's take a look at 'No pirates are sexy' again. I think the most natural way to read this is 'Not even one pirate is sexy', or a negation of 'A pirate is sexy'. Well, that's pretty easy – just take 'A (some) pirates are sexy', and slap a negation outside that bad boy:

¬Ex(Pirate(x) ∧ Sexy(x))

There's an important takeaway here. *You will never, ever see an English sentence translated as an existential conditional.* Take that, Sartre. Seriously though. It won't happen. think of it this way: It's really easy for an existential to be true. One thing in the domain has to satisfy it. One stinking thing. And think back to conditionals: on a truth table, they are true 3 out of four times. If the antecedent is false, the conditional is automatically true; if the antecedent is true, then the conditional is false when the consequent is false, but true when the consequent is true. So an existential conditional is really really easy to make true: all you have to do is find one thing in the domain that doesn't satisfy the antecedent.

Some Fs are Gs
Ex(F(x) ∧ G(x))

No Fs are Gs (alternative)
¬Ex(F(x) ∧ G(x))

Some Pirates are not Sexy

This, too, is nothing conceptually new. It's just a variation on some Fs are Gs, where Gs are negated.

Ex(Pirate(x) ∧ ¬Sexy(x))

Some Fs are not Gs
Ex(F(x) ∧ ¬G(x))

And those are the four Aristotelean forms. In the future, we'll ask whether a sentence is a Boolean, or an Aristotelean form (and which one).

Passivization and the forms

Much of the time, a sentence will wear its form on the surface. Especially if it's an Aristotelean form, the general rule is that if the English sentence begins with a quantifier word, that quantifier has the widest scope. Otherwise it's a Boolean. But sometimes the sentence doesn't have its quantifier all the way out front. Its form is buried. In some of these cases, there is nothing for it but to think hard about what the logic of the English sentence is. But there is at least one trick to help: passivizing.

Recall in English that passive and active voice sentences say exactly the same thing, they just say it slightly differently. So if you passivize, you're going to get a logically equivalent sentence. We can exploit this. Consider,

Albert pillaged every city

Hard to see what the structure is. For one, you can't even really see a connective – it's hidden. For another, that quantifier is hanging out in the middle of the sentence. Where does it go in the QL? Try passivizing. The passive voice reading of this is:

Every city was pillaged by albert

Aha! That's a bit more transparent, isn't it? It's clearly 'All Fs are Gs', or:

$\forall$x(City(x) $\rightarrow$ Pillaged-by-albert(x))

Now we can work on that complex predicate, 'Pillaged-by-albert'. Well it's just the plain old pillaging relation, with two argument spots, one for the pillager, the other for the pillaged. Who's the pillager? Our favorite arboreal rodent, Albert. Who's the pillaged? Every city. We're gonna use a pronoun for that, i.e. a variable. Is any previously mentioned variable restricted to cities? Yes, 'x'. So:

$\forall$x(City(x) $\rightarrow$ Pillaged(albert, x))

Remember. Even though the city comes before Albert in the English, it's still the thing pillaged, and thus goes in the *second* argument slot of 'Pillaged'.

Prepositions

Hoo boy. Back to grammar school.

Predicates modify individuals, we have that built in to our semantics. What do prepositions modify? Consider 'Every city between Tortuga and Port Royale has been sacked'. Is this a Boolean or an Aristotlean form? Well, that 'Every' all the way out front tells us it's a form (be careful: sometimes the English doesn't wear it's logical form on its sleeve like that, so later we'll go over some tricks for forcing sentences to do our will. But if the English quantifier is all the way out front, it's going to be a quantified sentence). In fact, it's 'All Fs are G', or

$\forall x(F(x) \rightarrow G(x))$

F and G in this case are themselves complex. F is 'being a city between Tortuga and Port Royale', and G is 'Has been sacked'. Okay, well G is not complex. So that means we can fill in the skeleton:

∀x(City between Tortuga and Port Royale(x) → Sacked(x))

How to break down the complex predicate 'being a city between Tortuga and Port Royale'? Well, first, what predicate terms are involved? 'Being a City' or more simply 'City', and the 3 place betweenness relation. What's between what? Every city (captured by the variable x, which is bound to that universal quantifier), and the two cities Tortuga and Port Royale. So what does the betweenness relation modify? That city x. So if x has two properties – it's a city, and it's between Tortuga and Port Royale – then it's sacked. How do we capture that x has these two properties? By conjoining them. Thus:

∀x((City(x) ∧ Between(x, tortuga, port royale)) → Sacked(x))

In this case. the preposition modifies the individual x, which we know is a city. But prepositions don't always modify individuals. Consider: Albert is a pirate, and a sexy one at that. We know pirates have a lover in every port, and because Albert is sexy, he does pretty well at this. So

Albert loves mary in tortuga and hanna in port royale

There is a reading of this where 'in Tortuga' modifies Mary, i.e., he loves Mary, and she's in Tortuga. But there's another reading where Albert, the cheating bastard, has one loving in Tortuga, and another in Port Royale. In this case, the preposition 'in [city]' modifies not the individual loved but the act of loving.

Perhaps this next example puts a finer point on it. Albert, fickle with his romantic attachments, loves Mary at 1 PM, and Manna at 2 PM. (it is unclear whether he keeps loving Mary at 2 PM, but Albert's impending relationship troubles are not our problem here. There's two predicates going on there – 'Loves', and 'At[time]' Does the at-a-time relation modify mary? It would be really odd to say that mary was 'at' 1 PM. Ditto for albert. Besides, that wouldn't capture Albert's fickleness – to say that

Loves(albert, mary) ∧ at(mary, 1PM)

doesn't at all capture what we want to say, which is that the loving itself is at 1PM. In this case, *the preposition modifies another predicate.* But how do we capture that? We don't have a mechanism for applying predicates to predicates!

It's a little complex, but we'll get the hang of it. The answer is that prepositions that modify predicates add another argument spot or arity to the predicate they modify. So ordinarily, 'Loves' is a two-place predicate. In this case, it's a 3-place predicate. What's the extra space for? It's for whatever is the subject of the preposition. In this case the preoposition is 'at[time]', and the subject is a time. So

Loves(lover, loved)

becomes

Loves-at(lover, loved, time of loving)

Remember, we're trying to translate 'Abert loves Mary at 1 PM and Hanna at 2 PM'. Is this a Boolean or a quantified sentence? There is no sign of any quantification anywhere – it's a Boolean. What kind of Boolean? The only Boolean word I see is 'and'; it's a conjunction. So the skeleton is:

(albert-loves-mary-at-1 ∧ albert-loves-hanna-at-2)

As we said, the prepositional phrase 'at[time]' modifies the loving, and we represent that by adding a new argument spot to 'Loves':

(Loves(albert, mary, 1) ∧ Loves(albert, hanna, 2))

Returning to 'Albert loves mary in tortuga and hanna in port royale', we get:

(Loves(albert, mary, tortuga) ∧ Loves(albert, hanna, port royale))

Take note, though: those are two different loving relationships. One holds between lovers, loved, and times, and the other holds between lovers, loved and places.

So make a lexicon stating this when you do translations like these.

Let's try one with some quantification:

Everyone Albert loves at 1 PM, he discards

What can I say? Don't fall in love with a pirate squirrel. At least not at 1 PM. (And really, if he's so callous about his 1 PM liasons, what makes you think he's any more faithul at other times of day?) Anyway, that sure looks like 'all Fs are Gs'. So the skeleton is:

∀x(L(x) → D(x)) or in more detail
∀x(Loves-at-1(x) → Discards(x))

We already discussed how to treat Loves-at-1:

∀x(Loves(albert, x, 1) → Discards(x))

Wait, how'd we get to Loves(albert, x, 1)? Simple. Loves, we said, was now a 3 place predicate, relating a lover, a loved, and a time. Who's the lover in this equation? Albert. So he goes in the first argument slot. Who is the loved? Everyone. (Or rather, everyone who is loved has a certain property, being discarded). Who is everyone? x is bound to the universal quantifier, so x goes in the second argument slot. And the time of loving? 1. As a rule of thumb, it is good practice when you are done with a translation – especially a complex one – to go back and make sure the right subjects are in the right argument spots by reasoning this way.

One last thing. That consequent isn't quite right. It would be really odd to say 'Albert discards'. I guess that could be a way of saying 'Albert is a user', but even that hides something important. You don't just use or discard, you use or discard something. 'Discards' is a two place relation, isn't it, involving a discarder and a discarded? So retaining our assumption of active voice, (i.e. 'Discards(x,y)' is 'x discards y' and not 'x is discarded by y', who is the discarder here? Yeah, it's that cheating bastard Albert. Who is discarded? Everyone Albert loved at 1. But we've already characterized those people in the antecedent – that's x. So,

∀x(Loves(albert, x, 1) → Discards(albert, x))

Variables bound to Multiple Quantifiers

In general, this is a no-no. But really, it's only incorrect when one variable is within the scope of multiple quantifiers. If you have a variable, say, 'x', and it's bound to two quantifiers *but no single instance is within the scope of both*, you're legit. Examples:

∃x∀x(Pirate(x) ∧ Squirrel(x)) : non-well-formed.
∃xPirate(x) ∧ ∃xSquirrel(x) : legit

I personally think that the legit sentence is bad practice translating, because it's confusing. As a rule of thumb, whenever I introduce a new quantifier, I bind it to a new variable. E.g.,

∃xPirate(x) ∧ ∃ySquirrel(y)

But this isn't a matter of logic, and it doesn't make any logical difference; it's just easier to read. Here's why. Think of variables, again, as being a lot like pronouns. Consider:

Albert pillaged a city. <u>It</u> burned. Then he pillaged another city. <u>It</u> burned too.

Think of 'it' as 'x'. In this case, the first x is bound to that first city, and the second x is bound to the second city. Even thought the same word ('it') is used twice, it refers to (potentially) different things. (Although there is no guarantee that they are the same city, it's not ruled out). Similarly, in the above "legit" example, that sentence tells us that there is a pirate and there is a squirrel. Maybe they are one and the same, maybe not. The referent of the variable is 'washed away' when the scope of its binding quantifier ends. And the variable, on its own, has no meaning – it just refers back to something else. So:

∃xPirate(x) ∧ ∃xSquirrel(x)
∃xPirate(x) ∧ ∃ySquirrel(y)

mean exactly the same thing. 'x' and 'y' both translate as 'it'. Nonetheless, I think the first sentence is a bit confusing in a way the second sentence is not, so as I said, I try to add new variables when I add new quantifiers.

Restricted Quantification

We've been cheating a bit. I have been repeatedly translating 'Everyone' as plain old '∀x'. But unless modified, that actually means 'everything'. And no matter how broad your concept of a person is, it seems difficult for me to imagine domains that only include persons. Let me give an example to drive the point home.

Suppose you knew this guy, call him Albert, and his sexual proclivities were so far ranging that you wanted to say he was sexually attracted to everyone. Men, women and everything else on the gender spectrum. You never saw him turn down an offer. Every body type, every personality type, etc. Okay, that's sort of unusual, but whatever. What if we translated this as

∀xAttractedTo(albert, x)?

That actually says that Albert is attracted to everything. Persons, goats, shoes – everything in the domain. But we have no reason to think that Albert goes home and fantasizes about a pair of Bruno Maglis or a petunia. And he's actually turned up his nose at the neighbor's sheep before. So this can't be right.

Can we simply say that we are restricting ourselves to domains consisting of only persons? That seems like a really inelegant, ad hoc solution. Plus, it won't work. What if you wanted to say that Albert is attracted to everyone, and he also makes shoes? Well you just stipulated that there are no shoes in the domain.

English has many of these 'restricted quantifiers':

Some/Every one
Sometimes
Some/Every where

Try to come up with more.

Each of these involves a quantifier and a category that the quantifier is restricted to. Above, it's persons, times and places, respectively and in that order. So think of a restricted quantifier as involving those two elements – the quantifier, and the category restricted to. Then the way to capture this is as an Aristotelean form. If it's a universal, it's 'all Fs are Gs', where F is the category. So:

Albert is attracted to everyone

First, we use our passivization trick to see if we can get that quantifier all the way out front

Everyone has Albert attracted to them

Skeleton:

$\forall x(\text{AttractedTo}(\text{albert}, x))$

(remember, even though we passivized, Albert is still the subject, and everyone the direct object)

But we want to restrict the objects of Albert's lust to persons. Easy-Peasy. If x is a person, albert is attracted to them:

$\forall x(\text{Person}(x) \rightarrow \text{AttractedTo}(\text{albert}, x))$

Notice this doesn't rule out Albert having shoe fantasies, but it doesn't commit him to them, either. Which is just as the English works. If I say that Albert is attracted to everyone, and someone says 'Hah, Smoove D, you lie! Albert also has strange fetishes about books!', well, have I lied, and have I been contradicted? No. I made no claims about anyone but people.

So in general, a quantified sentence of the form 'Every[category] is F' should be read as:

Every[Category] is F
$\forall x\ (\text{Category}(x) \rightarrow Fx)$

Keep in mind that F may itself be pretty complex. You might have

Everyone that Jeeves serves is a bachelor[9]

On the surface that looks like

$\forall x(\text{Serves}(\text{jeeves}, x) \rightarrow \text{Bachelor}(x))$

But 'everyone' just sounds like the people he serves. (Strange case. Who else would you serve? Maybe Jeeves is a devotee of the Old Gods and serves Cthulhu also. I don't know). So according to what I've just said, it's really

$\forall x(\text{Person}(x) \rightarrow (\text{Serves}(\text{jeeves}, x) \rightarrow \text{Bachelor}(x)))$

That works. I think the better way of putting it is probably the logically equivalent:

$\forall x((\text{Person}(x) \wedge (\text{Serves}(\text{jeeves}, x)) \rightarrow \text{Bachelor}(x))$

(recall from Boolean logic that $(P \rightarrow (Q \rightarrow R))$ is logically equivalent to $((P \wedge Q) \rightarrow R)$

The same reasoning applies to sometimes, someone, etc. But remember that there are no existential conditionals. Some Fs are Gs is a conjunction, $\exists x(F(x) \wedge G(x))$. So:

Some[Category] is F
$\exists x\ (\text{Category}(x) \wedge F(x))$

'Someone in the room farted'. If you say that, it's not as if you including this piece of chalk in your inquiry. You're claiming that there is a person in the room who farted.

$\boldsymbol{\exists x\ (\text{Person}(x) \wedge \text{InThisRoom}(x) \wedge \text{Farted}(x))}$

As we'll see, 'InThisRoom' is itself a fairly complicated predicate, and it involves the indexical 'This',

9 If you have never read PG Wodehouse's “Jeeves and Wooster”novels, you should. Or at least watch the BBC series.

which has a logic all its own, so we'll just gloss over the complexity here.)

Sometimes, Albert loves Mary
$\exists x(Time(x) \land Loves(albert, mary, x))$

That's a bit trickier. On the one hand, the 'sometimes' is all the way out front, so it's 'Some[person] is F'. One the other hand, 'Sometimes' is modifying the loving, so it's a prepositional phrase of that kind, which means it will add a third argument slot (for the time of loving) to the Loves relation.

This has been a non-exhaustive discussion of the logic of English. In many ways, Logic wants to grow into Formal Semantics, and provide a completely regimented treatment of natural language (or even unnatural languages, like English). This is just a start, or a 'fragment' as many semanticists would call it. There are all sorts of philosophically rich parts of English that cry out for such treatment – time, necessity and possibility, moral obligation – just to mention a few areas. But logic is the handmaiden of Philosophy, and not vice versa; these topics fall beyond the scope of this text.

With that in mind, let us Provinate.

Rules and Definitions

All Fs are Gs $\forall x(F(x) \rightarrow G(x))$

No Fs are Gs $\forall x(F(x) \rightarrow \neg G(x))$ **OR** $\neg Ex(F(x) \wedge G(x))$

Some Fs are not Gs $Ex(F(x) \wedge \neg G(x))$

Some Fs are Gs $Ex(F(x) \wedge G(x))$

Chapter V

Provinating

"Provinating the countryside! Provinating the peasants! Provinating all the people! In their THATCHED-ROOF COTTAGES! THATCHED-ROOF COTTAGES!" - Trogdor the Logic Dragon (https://www.youtube.com/watch?v=mm-aovm1axQ)

Hey, this is the fun part, right? All that earlier stuff was busywork leading up to this. Guess what: There are four new rules that comprise the proof system oh QL. Four stinking rules. There's two derived rules, but those just make things go a little quicker.

We're going to follow our usual practice and give each new connective / operator both a rule for introducing it, and a rule for getting rid of it. We'll start with the two easy ones – Existential Intro, and Universal Elim.

Existential Intro

If Albert is a pirate, well then surely s*omething* is a pirate.

Existential Intro
If you have
φα
on a line, you may write
∃xφx
on a subsequent line. Cite the line containing φα , and Existential Intro, as your justification.

Remember to not use a variable that is already bound within φ! This is called *quantifying over* α. Existentially Quantifying over it, to be precise. An example:

.
.
.
.
7) Fa → Ga
8) ∃x(Fx → Gx) 7, ∃ Intro
.
.
.

You don't need any new subproofs or anything like that. This is because if Albert meets a certain (in this case, conditional) condition, then it just follows automatically that something meets this condition.

This allows us a heuristic for thinking about existentials. In many ways they are like disjunctions. Really big disjunctions. An existential like ExPirate(x) is much like saying, 'either Albert is a pirate, or Hanna is a pirate, or Jeeves is a pirate, or. . .' But it's really big, because it covers everything in the domain (which may be infinite). Consequently, the existential rules work much like the disjunction rules. In particular, ∃ Intro works a lot like ∨ Intro. E intro allows you to move from a claim like 'Albert is a pirate', to the disjunction, 'Albert is a pirate, or Hanna is a Pirate, or. . . .'. Just like v Intro, it's kind of weird – you move from a state of knowing something for sure, to a state of knowing much less. You are essentially throwing away information. Why would you do that? Well, just like with ∨ Intro, sometimes you need to get something into a particular form, so you can work with it.

Sometimes this rule is called 'Existential Generalization'.

Universal elim

A universal is sort of like a law. It says that everything in the Domain meets some condition. Universal conditionals are especially like laws – they say that for everything in the domain, if you do F to it, it will G[10]. So if everything meets a condition, then for any random object you pick in the domain, it also must meet that condition:

Universal Elim
If you have
∀xφx
on a line, you may write
φα, for any individual **α** in the domain, on a subsequent line. Cite the line containing **∀xφx** , and Universal Elim, as your justification.

That line φα is sometimes said to be an *instance* of the universal ∀xφx. Hence this rule is sometimes referred to as Universal Instantiation. Here's an example:

.
.
.
7) ∀x(Fx → Gx)
8) Fa → Ga   7, ∀ Elim
.
.
.

Again, you don't need anything fancy – no subproofs or nothin'. This is because 'Everything' means 'Everything', and that is guaranteed to cover every individual in the domain. If everything that is a pirate is sexy, then it's just automatic that if Albert is a pirate, he's sexy.

Again, this allows us a heuristic for thinking about universals. If they cover every individual in the domain, then they cover this thing, and this thing, and this thing, and. . . So in effect they are like conjunctions containing every individual as part of a conjunct. If everything is a pirate, then Albert is a pirate, Jeeves is a pirate, Jodney is a pirate. . . .and so on for every individual. And the Universal Elim

10 In fact, the similarity between universal conditionals and laws is so striking, that many philosophers have thought that laws of nature are simply all universal conditionals.

is really analogous to Conjunction Elim.

Also notice that just like conjunction elim, you are allowed to instantiate a universal as many times as you want. In fact, some proofs may require you to instantiate the same universal multiple times (these proofs tend to be very difficult!) This is one area of QL that requires some art and insight. You might have to think quite hard about what individuals you want to instantiate your universal. They will all be true, but only some may be useful to you. It is possible to instantiate a universal with some individual that is useless to you, but you do not realize it and you get stuck in a proof.

Just two more rules! 'Smoove!' you say, 'This is cake! I got this puppy wired for sound! Easy-Peasy!' Heh. Heh. Heh. Now the fun starts.

Universal Intro

Wait. How can you possibly *prove* a universal? I mean, you can do induction (The sun rose every day for the last 45 days, thus I predict it rises tomorrow, and for that matter, every day), but that's not *proof* . It's not certain, for one thing.

But what if you started with some universal information? What if you had some kind of universal law, for example? Shouldn't you be able to derive other universal laws from that? Seems likely. And that insight is at the heart of this particular rule. But it is tricky.

First, what we can do is suppose – that means starting a subproof, by the way – that we have some kind of very special object. I call this “the universal object”. It manages to be everything all at once. But we start off knowing absolutely nothing about it. In particular, we cannot identify it with any particular object. (See what I did there?) If any objects have been mentioned in the proof already, then we know something about those objects. So they can't be the universal object. When we begin our subproof, we name a universal object, and it has to be completely arbitrary. It cannot be any object that's been mentioned in the proof before. So what we do is just pick an object at random, and put a little box around it. The box is not a logical requirement, it's just there to remind us that this object has to meet certain conditions – it's wholly arbitrary, that is, it's never been mentioned in the proof before. Let's call it albert, and suppose that it meets the arbitrariness requirement.

Then we start to provinate within the subproof. But how can we say anything about the universal object? We don't know anything about it! After all, the arbitrariness requirement means it's never been mentioned in the proof yet!

There is one, and only one way, that we can say anything about Albert. That is, we can only make claims about Albert if we could make those claims about anything in the domain at all. That is, if we had some universal information that covered Albert even if it didn't mention him in particular. Let's build our reasoning:

1)∀x(Pirate(x) → Squirrel(x))
2) ∀x(Squirrel(x) → Sexy(x))
∴∀x(Pirate(x) → Sexy(x))

Pretty straightforward, right? All pirates are squirrels, all squirrels are sexy, so all pirates are sexy. Now let's pick some totally arbitrary individual, one that has never been mentioned in the proof and one whom we know nothing about, and have her be our universal object.

1)$\forall x(Pirate(x) \rightarrow Squirrel(x))$	Premise
2) $\forall x(Squirrel(x) \rightarrow Sexy(x))$	Premise
hanna	
n) $\forall x(Pirate(x) \rightarrow Sexy(x))$	

Notice how we have 'flagged' Hanna. That's just to remind ourselves that Hanna must be arbitrary, i.e., not appear anywhere earlier. Hanna clearly meets that requirement.

Now if Hanna is really the universal object, anything we say about Hanna, we can say about everything. That is, it will hold for all x. What is it that, in this case, we want to hold for all x? To answer that, look at what our goal is (the conclusion). We want to show that all individuals have this property: if they are pirates, they are sexy. That is, we want to show that all x's meet the condition,

$(Pirate(x) \rightarrow Sexy(x))$

If we can show that Hanna, the universal object, meets that, then it follows that everything meets that. So our proof looks like:

1)$\forall x(Pirate(x) \rightarrow Squirrel(x))$	Premise
2) $\forall x(Squirrel(x) \rightarrow Sexy(x))$	Premise
3) hanna . . . n-1) $Pirate(hanna) \rightarrow Sexy(hanna)$	
n) $\forall x(Pirate(x) \rightarrow Sexy(x))$	$\forall$ Intro, 3-(n-1)

The final move there is the ∀ intro. We are saying that IF we assume hanna is our universal object, and we can prove Pirate(hanna) → Sexy(hanna), then we know that that conditional actually holds universally, for every x. The proof's not done yet, though. We need to get from 1-3 to (n-1). That's going to be hard – nothing mentions Hanna, so how can we prove anything about her? Actually, the two premises don't mention Hanna directly, but they are about everything in the domain – *including Hanna*! So let's universal elim x in favor of Hanna, in 1 and 2:

1)∀x(Pirate(x) → Squirrel(x))	Premise
2) ∀x(Squirrel(x) → Sexy(x))	Premise
3) hanna	
4) Pirate(x) → Squirrel(x)	∀ elim 1
5) Squirrel(x) → Sexy(x)	∀ elim 2
.	
n-1) Pirate(hanna) → Sexy(hanna)	
n) ∀x(Pirate(x) → Sexy(x))	∀ Intro, 3-(n-1)

Now what is left is a straightforward Boolean proof. This illustrates the first and biggest point of proof strategery in QL: what you do is strip off the quantifiers from the premises, set up the endgame to build any new quantifiers in, and then in the middle just do an ordinary Boolean proof. In our case we could proceed in two ways, a fast but memory-intensive one (because you have to remember a new rule), or a slow but low-memory one (because you basically prove the previous rule on the fly). Here's the fast version:

1)∀x(Pirate(x) → Squirrel(x))	Premise
2) ∀x(Squirrel(x) → Sexy(x))	Premise
3) hanna	
4) Pirate(x) → Squirrel(x)	∀ elim 1
5) Squirrel(x) → Sexy(x)	∀ elim 2
6) Pirate(hanna) → Sexy(hanna)	Hypothetical Syllogism 4,5
n) ∀x(Pirate(x) → Sexy(x))	∀ Intro, 3-(n-1)

Boom, done. The only problem is that you have to memorize Hypothetical Syllogism and see the pattern. Here's the other way. Notice that your immediate goal is a conditional,

Pirate(hanna) → Sexy(hanna)

So you have to prove a conditional. Set up a Conditional Intro subproof:

1)∀x(Pirate(x) → Squirrel(x))	Premise
2) ∀x(Squirrel(x) → Sexy(x))	Premise
3) hanna	
4) Pirate(hanna) → Squirrel(hanna)	∀ elim 1
5) Squirrel(hanna) → Sexy(hanna)	∀ elim 2
6) Pirate (hanna)	assume
.	
.	
n-2) Sexy (hanna)	?
n-1) Pirate(hanna) → Sexy(hanna)	→ Intro
n) ∀x(Pirate(x) → Sexy(x))	∀ Intro, 3-(n-1)

Cake from here on out. We just daisy-chain along with a few → elims and we are done:

1) ∀x(Pirate(x) → Squirrel(x))	Premise
2) ∀x(Squirrel(x) → Sexy(x))	Premise
3) hanna	
4) Pirate(hanna) → Squirrel(hanna)	∀ elim 1
5) Squirrel(hanna) → Sexy(hanna)	∀ elim 2
6) Pirate (hanna)	assume
7) Squirrel (hanna)	→ elim 4,6
8) Sexy (hanna)	→ elim 7,5
9) Pirate(hanna) → Sexy(hanna)	→ Intro 6-7
10) ∀x(Pirate(x) → Sexy(x))	∀ Intro, 3-9

Same proof, really.

Okay, time for some post-game analysis. Let's face it, there was nothing new in 4-9 (or 4-6 in the Hypothetical Syllogism version). All the action is going down in the assumption (line 3) and the last two lines. Everything else is just a boring Boolean proof. What's really happening is that we let Hanna stand in for 'everything', and then put her in the subject position. But because Hanna never appears in the proof earlier, there's no way to say anything about her – unless you can say it about anything. Luckily, the two premises are universal as well. So there is a lot we can say about Hanna, even if we know nothing else about her. We can say it because 1 and 2 say things about everyone, including Hanna. At the end of the subproof, we've only made claims that would apply to everyone in the domain. If they apply to everyone in the domain, we can universally quantify over Hanna, who was really just a stand-in for 'everything'.

Here's another way of putting it. What if you had picked another individual in (3) as your universal object? Say, Wooster. Could you have proven

Pirate(Wooster) → Sexy(Wooster) ?

Of course. We would simply have instantiated 'wooster' for 'x' in lines 4 and 5. What if we had chosen Jeeves? Same thing. Think of the domain as a magician top hat, with a bunch of names on slips of paper in it. No matter who you draw out of the hat, you will be able to ∀ elim 1 and 2 for that individual. And so you will be able to show that if the are a pirate, they are sexy. Try it! That's the essence of ∀ elim – if you can prove that some arbitrary object has a property, then that could only follow from universal information, so it must hold of anything.

The rule:

∀ Intro
If you assume, on the first line of a subproof, that **α** is a universal object, then if you derive **φα**, you may exit the subproof and write **∀xφ(x)**.
As justification, cite the entire subproof.
α must never have appeared in the proof before, which is what makes it arbitrary and a universal object.

The proof form:

.
.
7)
hanna
.
.
.
12) Pirate(hanna) → Sexy (hanna)

13) ∀x(Pirate(x) → Sexy(x)) ∀ Intro 7-12

The flagged individual, Hanna, must be arbitrary (i.e. never mentioned before).

Existential Elimination (∃ Elim)

This one is also a bit tricky. Suppose you know that something is a pirate, but you don't know who it is. What can you really say about that pirate, in addition? Not much – unless it's stuff you could say about pirates in general. In other words, just as in ∀ intro, you can't really say much about them unless it follows from some universal information.

The first thing to do is to strip off the quantifier so that we can work with it. Remember, we don't want to say anything about our pirate unless it's true of all pirates. So we have to be careful what name we use to replace 'Something'. Again we take a hint from ∀ intro, and pick an individual that we have never mentioned in the proof before. We flag it, because it must be arbitrary (never appeared before) and temporary (doesn't come out of the subproof). It's just a dummy term, a kludge for getting 'something' into the subject position:

1) ∀x(Pirate(x) → Pillages (x, tortuga))	Premise
2) ∃xPirate(x)	Premise
3) wooster Pirate(wooster)	Assumption
4) .	
n) ∃xPillages(x, tortuga)	?

Line 3 is the result of replacing 'something' in line 2 with wooster. So it looks like we have eliminated the E in line 3. But notice *we do not cite the rule, nor do we cite line 2, here.* Why not? Many people find this confusing. It's because our flagged name, Wooster, must be both arbitrary and temporary. And we will not have shown that it is temporary until the end of the proof.

Here's another way to think about it. Recall that we likened existentials to big disjunctions. The rule for getting rid of disjunctions is ∨ elim, sometimes called Argument by cases or AC. In ∨ elim, we run through subproofs involving each disjunct, and only at the end, when we have exhausted all the disjuncts, are we justified in eliminating the ∨. ∃ elim is similar, only because our individual is arbitrary and temporary, it stands in for all the disjuncts at once.

E elim tells us that if our choice of name is arbitrary and temporary, we can remove the last line from the subproof. What would we like that last line to be, then? Easy – what are we aiming for? In this case, we are trying to aim for the conclusion,

∃xPillages(x, tortuga)

So, setting up our subproof,

1) ∀x(Pirate(x) → Pillages (x, tortuga))	Premise
2) ∃xPirate(x)	Premise
3) wooster Pirate(wooster)	Assumption
4)	
.	
.	
n-1) ∃xPillages(x, tortuga)	?
n) ∃xPillages(x, tortuga)	∃ elim 2, 3 - (n-1)

Notice, again just like ∨ elim, the last two lines are identical. This will always be the case. As our justification for line n, we cite both the line that we eliminated the ∃ from, and the subproof that a) shows our choice was arbitrary and temporary and b) ends with the line we wish to take out of the subproof. Remember that removing a line from a subproof is saying that the line does not depend on the assumption.

From here on, the proof is simple. We just instantiate the Universal in premise one, and then it's ordinary Boolean logic:

1) ∀x(Pirate(x) → Pillages (x, tortuga))	Premise
2) ∃xPirate(x)	Premise
3) wooster Pirate(wooster)	Assumption
4) Pirate(wooster) → Pillages(wooster, tortuga)	A Elim 1
5) Pillages(wooster, tortuga)	→ elim 3,4
6) ∃xPillages(x, tortuga)	?
7) ∃xPillages(x, tortuga)	∃ elim 2, 3 - (n-1)

Why did we choose to instantiate 1 with Wooster? Easy: he's the only person we care about in this proof (because of 3). And how do we get from 5 to 6? Well, are we introducing a symbol in 6, or getting rid of one? We are introducing one – an existential ∃. So we will use E intro. This does two things: it gets us the result in (6) that we want, and it gets rid of the flagged name Wooster, making the choice of Wooster in (3) temporary. It's also arbitrary, so our use of E elim in (7) is legit.

A note on the use of ∃ Intro in (6): the penultimate step of our proof will almost always involve getting rid of the flagged name, "the scarlet letter", as I call it – because the flagged name that must be arbitrary and temporary. A very common way to do that is via ∃ intro, since that essentially replaces the flagged name with a variable (bound to an existential quantifier). The other major way to do it is to have a Reductio subproof nested inside the E ∃lim. The last line of this will be ⊥, a contradiction, which has no names in it. You will then be able to pull ⊥ out of the subproof, often very useful. Together these make up the vast majority of penultimate lines.

1) ∀x(Pirate(x) → Pillages (x, tortuga))	Premise
2) ∃xPirate(x)	Premise
3) wooster Pirate(wooster)	Assumption
4) Pirate(wooster) → Pillages(wooster, tortuga)	A Elim 1
5) Pillages(wooster, tortuga)	→ elim 3,4
6) ∃xPillages(x, tortuga)	∃ Intro 5
7) ∃xPillages(x, tortuga)	∃ elim 2, 3 - (n-1)

The rule, formally:

∃ Elim

If you have a sentence of the form **∃xφx**, you may form a new subproof beginning with **φα** for any a in the domain.

Any sentence that follows from **φα** may be taken outside the subproof as long as **α** is arbitrary (does not appear prior to the subproof) and temporary (does not appear in the line being removed from the subproof)

The form:

1) .	
2) ∃xφx	
3) wooster φ(wooster)	Assume
4) .	
5) .	
6) ψ	
7) ψ	∃ elim 2, 3 – 6 (where wooster is arbitrary and temporary)

That is all four rules.

Some Mistakes

There are a few common mistakes. Students often forget which rules require subproofs (It's ∃ elim and ∀ intro). They also forget that in ∀ intro, you begin the subproof with a naked name only, while in E elim, you begin by replacing the x's in some existential sentence with a name, and giving that new sentence. But the most common mistake is when people forget to make the flagged name arbitrary and/or temporary. So let's look at an example, because I think it will show that the rules really do work the way they are supposed to – they won't let you prove universals that don't really follow, for example.

My wife reads all these cheesy serial killer novels. She especially loves the ones with psychological profilers. As such, I've picked up some cheesy serial killer facts over the years. One is that CSKs almost always kill within their own race. Who knew? So here we are, beat cops in London 1888. Someone is killing the white prostitutes. One is named Catherine Eddowes. So we know some facts.

∃xKills(x, eddowes)
White(eddowes)
∀x (Kills (x, eddowes) → SameRace(x, eddowes)

There's a right way and a wrong way this proof could go. Let's start with the right way:

1) ∀x (Kills (x, eddowes) → SameRace(x, eddowes)	Premise
2) ∃xKills(x, eddowes)	Premise
3) White(eddowes)	Premise
4) jack Kills(jack, eddowes)	Assumption
5) (Kills (x, eddowes) → SameRace(x, eddowes)	A Elim 1
6) SameRace(jack, eddowes)	→ elim 4,5
7) Kills(jack, eddowes) ∧ SameRace(jack, eddowes)	∧ Intro 4, 6
8)∃x Kills((x, eddowes) ∧ SameRace(x, eddowes))	E Intro 7
9) ∃x Kills(x, eddowes) ∧ SameRace(x, eddowes)	∃ elim 3, 4-8

The proof went almost exactly the way our initial example proof went. We didn't do anything with the fact that eddowes was white, but with a little extra work, we could use that to show that eddow's killer was white. Our conclusion, something exists, and it killed eddowes and is the same race as her, basically says 'the killer is the same race as Eddowes'. So we have narrowed some things down, and now the police know, at least, that they should be looking for a white guy. And we did this by temporarily assigning the killer the name 'jack' (Jack the Ripper, of course). This is what the police did – they didn't actually think he name was Jack, they just called him that temporarily so they could talk about him.

Suppose, however, that a beat cop had been looking in the phone book, and noticed that a guy named Jack lived at 121 Drury Lane. So we now know

Lives(jack, 121dl)

Can we take some crime-solving, logic-chopping shortcuts?

1) ∀x (Kills (x, eddowes) → SameRace(x, eddowes)	Premise
2) ∃xKills(x, eddowes)	Premise
3) lives(jack, 121dl)	Premise
4) jack Kills(jack, eddowes)	Assumption
5) (Kills (x, eddowes) → SameRace(x, eddowes)	A Elim 1
6) SameRace(jack, eddowes)	→ elim 4,5
7) Kills(jack, eddowes) ∧ lives(jack, 121dl)	∧ Intro 4, 6
8)∃x Kills((x, eddowes) ∧ lives(jack, 121dl))	E Intro 7
9) ∃x Kills((x, eddowes) ∧ lives(x, 121dl))	∃ elim 3, 4-8

Fantastic work! The killer lives at 121 Drury Lane! Send the SWAT team! Boy, that murderous bastard is going to be surprised! Wait. . huh, what? That can't be right! It's way too easy! What's more, we could use that method to prove all sorts of total rubbish. What went wrong?

We didn't pick an arbitrary object in (4). We picked jack, but he's already mentioned in the proof. So it's not the case that anything you say about jack, you can say about anyone. What we say about jack here depends on some very specific information we had about him. So the name is not being used simply as a dummy term. This illustrates the importance of making your flagged names arbitrary and temporary!

Of course, the rules are the easy part. What is hard is strategery. So let's go over that.

Rules and Definitions

Existential Intro
If you have
φa
on a line, you may write
∃xφx
on a subsequent line. Cite the line containing φα , and Existential Intro, as your justification.
Example:

.
7) Fa → Ga
8) ∃x(Fx → Gx) **7, ∃ Intro**
.

Universal Elim
If you have
∀xφx
on a line, you may write
φα, for any individual **α** in the domain, on a subsequent line. Cite the line containing **∀xφx** , and Universal Elim, as your justification.

.
7) ∀x(Fx → Gx)
8) Fa → Ga **7, ∀ Elim**
.

∀ Intro

If you assume, on the first line of a subproof, that **α** is a universal object, then if you derive **φα**, you may exit the subproof and write **∀xφ(x)**.

As justification, cite the entire subproof.

α must never have appeared in the proof before, which is what makes it arbitrary and a universal object.

.
.

7)

hanna

.
.
.

12) Pirate(hanna) → Sexy (hanna)

13) ∀x(Pirate(x) → Sexy(x)) **∀ Intro 7-12**

∃ Elim

If you have a sentence of the form **∃xφx**, you may form a new subproof beginning with **φα** for any a in the domain.

Any sentence that follows from **φα** may be taken outside the subproof as long as **α** is arbitrary (does not appear prior to the subproof) and temporary (does not appear in the line being removed from the subproof)

1) .
2) ∃xφx

3)

wooster

φ(wooster) Assume
4) .
5) .
6) ψ

7) ψ **∃ elim 2, 3 – 6 (where ψ does not contain α)**

Chapter 6

Proof Strategery and Derived Rules

Ah, strategery. We've already hinted at what the basic strategery is: get rid of the quantifiers using elim rules, do the subsequent boolean proofs, and build the quantifiers back up. Rip 'em down, proove, build 'em up. What does that look like in more detailed terms? We now introduce the Belcher Pentapartite Strategery:

Phase one: Is the conclusion a Universal? Set up a ∀ Intro proof. Is there an Existential in the premises? Set up an existential elim proof. Are there negated quantifiers in your premises? Use the Quantifier Exchange rules to distribute them. RAA.

Phase two: Instantiate any Universals in your premises.

Phases 3-5: As per the Tripartite strategery. Build up any Existentials you need using ∃ Intro.

(See the appendix for a handy and detailed version of this)

Damn, that was easy. 'But Smoove!' you howl. 'What's this quantifier exchange rubbish?'. Oh yeah, we'd better cover that. Those are derived rules.

The insight to QE is this. If everything is a pirate, nothing fails to be a pirate. And if something is a pirate, then not everything is a non-pirate. So:

$\neg\forall\neg\varphi x \Leftrightarrow \exists x\varphi x$

$\neg\exists\neg\varphi x \Leftrightarrow \forall x\varphi x$

Let's prove it. Just like DeMorgan's, these are quite difficult.

1) ¬∀¬Pirate(x) . . . n) ∃xPirate(x)

Not much to do directly, here. We don't have any rules for distributing those negations – in fact, that's what we are trying to prove. And we have no rules for dealing with negated quantifiers. I'm stumped. What do you do when you're stumped? Set up RAA!

1) ¬∀¬Pirate(x)	Premise
2) ¬∃xPirate(x)	Assume for RAA
.	
.	
.	
cthulhu)⊥	
n) ∃xPirate(x)	RAA 2 - cthulhu

Wait, what's this cthulhu business? Well, we need some way of labeling lines that we haven't gotten to yet, and hence don't have a line number for. I've been using n, n-1, and so on. But I hate that. It's so pretentious. So I'm am going to label those lines with fun stuff. I default to using the names of cthulhu mythos deities[11]. But you can use all sorts of things. Cocktails, mayors, animated TV shows, whatever.

Anyway, we are still in a bind. Now we've got a negated existential in addition, and we still have our same problem with the negation. We need to generate some contradiction – what's around to

11 Don't know what the Cthulhu mythos is? Run, don't walk, and get a book of H.P. Lovecraft short stories.

contradict? Only our premise, ¬∀¬Pirate(x). Its negation is ∀¬Pirate(x):

1) ¬∀¬Pirate(x)	Premise
2) ¬∃xPirate(x)	Assume for RAA
.	
.	
yog-sothoth) ∀¬Pirate(x)	?
cthulhu)⊥	⊥ Intro, 1, yog-sothoth
n) ∃xPirate(x)	RAA 2 - cthulhu

Okay now we are getting somewhere. Even if it doesn't feel like it. We want to prove a universal, so we need to set up a ∀ Intro (phase 1 of the strategery)

1) ¬∀¬Pirate(x)	Premise
2) ¬∃xPirate(x)	Assume for RAA
3)	Assume for ∀ Intro
albert	
.	
.	
hastur) ¬Pirate(albert)	
yog-sothoth) ∀¬Pirate(x)	∀ Intro 3 - hastur
cthulhu)⊥	⊥ Intro, 1, yog-sothoth
n) ∃xPirate(x)	RAA 2 - cthulhu

Akkk! We still don't have anything new to work with! How are we gonna prove ¬Pirate(albert)? I am stumped again. RAA Time? We are in so many scope lines already! But every time we've added a scope line, we've said how we plan to get out. We have a clear exit strategy. So:

1) ¬∀¬Pirate(x)	Premise
2) ¬∃xPirate(x)	Assume for RAA
3)	Assume for ∀ Intro
albert	
4) Pirate(albert)	Assume for RAA
.	
.	
nyar) ⊥	⊥ Intro ?
hastur) ¬Pirate(albert)	RAA 4 - nyar
yog-sothoth) ∀¬Pirate(x)	∀ Intro 3 - hastur
cthulhu)⊥	⊥ Intro, 1, yog-sothoth
n) ∃xPirate(x)	RAA 2 - cthulhu

That's 'nyar' for 'Nyarlathotep', by the way. We'd better finish this soon, because I am running out of cthulhu deities, and you can't mention hastur more than twice (or he appears, much to the detriment of your sanity). Oh damn. I have mentioned him at least three times now. No wonder this is so crazy.

But look around. We are just a step away from a contradiction. Albert is a pirate, but (2) tells us nothing is a pirate. So let's just ∃ intro over Albert in (4), and we have our contradiction:

1) $\neg\forall\neg$Pirate(x)	Premise
2) $\neg\exists$xPirate(x)	Assume for RAA
3)	Assume for $\forall$ Intro
albert	
4) Pirate(albert)	Assume for RAA
5) $\exists$xPirate(x)	$\exists$ Intro 4
6) $\bot$	$\bot$ Intro 2,3
7) $\neg$Pirate(albert)	RAA 4 - 6
8) $\forall\neg$Pirate(x)	$\forall$ Intro 3 - 7
9)$\bot$	$\bot$ Intro, 1, 8
10) $\exists$xPirate(x)	RAA 2 - 9

Damn, that was rough. I promise you that was one of the harder proofs. Remember that it's the proofs where you start with so little, that are the hardest. Wait – we're only halfway there. We need to prove it in reverse. This will be a lot easier, I swear.

1) ∃xPirate(x)	Premise
.	
.	
.	
10)¬∀¬Pirate(x)	?

Well. We could start one of two ways. We could get rid of that existential with ∃ elim, or we could notice that our conclusion is a negation, so RAA would be useful. We're going to have to use both strategeries, so it really doesn't matter. Let us begin with ∃ elim:

1) ∃xPirate(x)	Premise
2)	
jeeves	
Pirate(jeeves)	Assume for ∃ elim
.	
.	
LW) ¬∀¬Pirate(x)	?
mart)¬∀¬Pirate(x)	∃ Elim, 1, 2 - LW

The new theme is cocktails. 'mart' is a Martini, 'LW' is a Last Word. Great drink – look it up. We've set up our ∃ elim, we know how we are going to get out of the subproof, and our choice of Jeeves is arbitrary and temporary. So we are on the right path.

What now? We can't do much with the fact that Jeeves is a pirate. Stumped again. (You will see a lot of this). All I can think of is another reductio:

1) ∃xPirate(x)	Premise
2)	
jeeves	
Pirate(jeeves)	Assume for ∃ elim
3) ∀¬Pirate(x)	Assume for RAA
.	
.	
GT) ⊥	⊥ Intro ?
.	
LW) ¬∀¬Pirate(x)	RAA 3 - GT
mart)¬∀¬Pirate(x)	∃ Elim, 1, 2 - LW

GT is, of course, the classic, never out of style, Ginnus Tonnicus. So now we are looking for a contradiction. (3) tells us that Everyone is a non-pirate. How sucky. But 2 tells us Jeeves is a pirate. Boom. Let's just ∀ elim (3) in favor of Jeeves. After all, if everyone is a non-pirate, so is Jeeves.

1) ∃xPirate(x)	Premise
2)	
jeeves	
Pirate(jeeves)	Assume for ∃ elim
3) ∀¬Pirate(x)	Assume for RAA
4) ¬Pirate(jeeves)	∀ elim 3
5) ⊥	⊥ Intro 2,4
6) ¬∀¬Pirate(x)	RAA 3 - 5
7)¬∀¬Pirate(x)	∃ Elim, 1, 2 - 6

It's a rule now. This is the first half of the Quantifier exchange rule, QE. The second half will be proving ¬∃¬φx ⟺∀xφx.

¬∃¬φx . . . ∀xφx

This looks a little more straightforward. Phase one of the strategery says to set up ∀ Intros, Strip off existentials with ∃ elim, or set up RAA. Well, we have to prove a Universal, so it's ∀ intro time.

1) ¬∃x¬Pirate(x)	Premise
2) jeeves . . slayer) Pirate(jeeves)	 ?
anthrax) ∀xPirate(x)	∀ Intro 2 - slayer

Ahh, Jeeves, the Universal Butler. If we can show he's a pirate, we can show that anything is a pirate. Our theme is thrash metal.

Hmm. We can't do anything with line 1, because we have no rule yet for distributing negations over existentials. Dang. I am stumped. What do you do when you're stumped? (This should be automatic by now!) Set up a reductio!

1) ¬∃¬Pirate(x)	Premise
2) jeeves	
3) ¬Pirate(jeeves)	Assume for RAA
.	
.	
exodus) ⊥	⊥ Intro ?
slayer) Pirate(jeeves)	?
anthrax) ∀xPirate(x)	∀ Intro 2 - slayer

As always, when you set up a RAA, at every new line you looks around for possible contradictions. The only things so far that could contradict are (1) and (3). But they aren't quite contradictions. Hey wait, though, if Jeeves is a pirate, then something is a pirate! That's a contradiction!

1) ¬∃x¬Pirate(x)	Premise
2) jeeves	
3) ¬Pirate(jeeves)	Assume for RAA
4) ∃x¬Pirate(x)	∃ Intro 3
5) ⊥	⊥ Intro 1,4
6) Pirate(jeeves)	RAA 3 - 5
7) ∀xPirate(x)	∀ Intro 2 - 6

Notice that in the move from 3 to 4, the ∃x goes *all* the way out front. This is always the case, if it wasn't apparent from the rules. So in sum: we assumed that Jeeves was our universal object. Since he's totally arbitrary, the only way we can say anything about Jeeves is if we could say it about anything. But if we could say it about anything, we could say it about ∀x. Ideally, we want to show that Jeeves is a pirate, because our goal is to show that everything is a pirate (on the assumption that nothing is a non-pirate). But then we're a bit stumped, because line 1 is our only complete sentence, and we have no rules for distributing the negations there (in fact, that's the rule we're trying to prove). So we set up an RAA. As soon as we set that up, we see that a contradiction is right nearby, so Jeeves – the universal butler – must in fact be a pirate. Done! Now in reverse:

∀xφx
.
.
¬∃¬φx

Hey, this one looks kind of easy! You want to strip off the universal because it's easy – *don't do that yet*! Save the ∀ elims for as late as possible. I will explain why in a minute. Instead, go for a reductio:

1)∀xPirate(x)	Premise
2)∃x¬Pirate(x)	Assume for RAA
.	
.	
cthulhu) ⊥	⊥ Intro
¬∃¬Pirate(x)	RAA 2 - cthulhu

You might think that you could pull a ∀ elim on 1, get Pirate(jeeves), and then ∃ intro, to get ∃xPirate(x). You could, but it wouldn't help. That does not contradict line 2. Together, they say, something is a pirate, and something is not a pirate. No contradiction there! Again, keeping in mind my advice to save ∀ elim for as late as possible, let's set up an ∃ elim of line 2:

1)∀xPirate(x)	Premise
2)∃x¬Pirate(x)	Assume for RAA
3) jeeves Pirate(Jeeves)	Assume for ∃ elim
.	
.	
nyar) ⊥	⊥ Intro
cthulhu) ⊥	∃ elim, 2, 3 - nyar
¬∃¬Pirate(x)	RAA 2 - cthulhu

Remember that the last line of the ∃ elim subproof will be the line we want to bring out of this subproof, and it must not have the Scarlet Letter in it (the flagged name at the beginning of the subproof). It's a ⊥, so that meets our requirements (remember I said that ∃ intro and ⊥ would be the two main ways of getting rid of the Forbidden Name?) But that means we have to change line nyar. That will be our ∃ elim. We finally showed that if we assumed our non-pirate was Jeeves, we'd get a contradiction; but sinceJeeves is arbitrary and temporary, it doesn't matter who we picked, we'd have gotten a contradiction (except we haven't shown the contradiction yet). So our contradiction does not depend on the choice of Jeeves and is thus not within that dependency line. Our contradiction is pretty quick:

1)∀xPirate(x)	Premise
2)∃x¬Pirate(x)	Assume for RAA
3) jeeves ¬Pirate(jeeves)	Assume for ∃ elim
4) Pirate(jeeves)	∀ Elim 1
5) ⊥	⊥ Intro 3, 4
6) ⊥	∃ elim, 2, 3 - 5
7)¬∃¬Pirate(x)	RAA 2 - 6

To recap: We assumed that something was a non-pirate, hoping to generate a contradiction and thereby prove that nothing was a non-pirate. Then we named that non-pirate a dummy term, arbitrary and temporary, 'Jeeves'. But (1) tells us everything is a pirate, so our assumption that Jeeves was a non-pirate means that he is both a pirate and a non-pirate. So he can't be a nonpirate. and since it was a dummy term, nothing could be a non-pirate.

QE is a rule of replacement, and pretty potent. Keep double negation rules in mind when you use it. For example, suppose you have a sentence like

¬<u>∀x</u>Pirate(x)

QE tells us we can replace that with

¬<u>¬∃x¬</u>Pirate(x)

and then we can use 2NE to get
∃x¬Pirate(x)

So in effect, QE is a rule for driving negations inside a sentence. Don't forget this!

Stripping off Quantifiers in the Right Order

Recall I have repeatedly said to wait until the last minute to do ∀ elim. Always do your ∃ elims first (or ∀ intros). In fact, if you have Universals embedded in your conclusion, say,

P → ∀xP(x),

or existentials embedded in your premises, say,

P → ∃xP(x)

Then try to do get to those as soon as possible. (Sometimes that may mean you *have* to do an ∀ elim first.) Here's why. Lets' go back to the last proof:

1)∀xPirate(x)	Premise
2)∃x¬Pirate(x)	Assume for RAA
nyar) ⊥	⊥ Into, ?
cthulhu)¬∃¬Pirate(x)	RAA 2 - nyar

Suppose at 3, we succumbed to temptation, and did an A elim on (1):

1)∀xPirate(x)	Premise
2)∃x¬Pirate(x)	Assume for RAA
3) Pirate(jeeves)	∀ elim 1
nyar) ⊥	∃ elim, 2, 3 - 5
cthulhu)¬∃¬Pirate(x)	RAA 2 - nyar

Now at some point (like now), we have to get rid of that existential in line 2:

1)∀xPirate(x)	Premise
2)∃x¬Pirate(x)	Assume for RAA
3) Pirate(jeeves)	∀ elim 1
4) jeeves ¬Pirate(jeeves)	Assume for ∃ elim
nyar) ⊥	∃ elim, 2, 3 - 5
cthulhu)¬∃¬Pirate(x)	RAA 2 - nyar

But right away at line (4) we are in trouble. Jeeves is not arbitrary! He appears in (3)! In fact, we can pick *any* individual *except* Jeeves! Since we cannot use Jeeves here, we can't generate the contradiction we want. In fact, we can't make line (4) interact with any other premises! (Remember what happens when you pick a non-arbitrary name for your ∃ elim).

Always try to do your ∃ elims first, so that they will be arbitrary and *then* you can instantiate your universals so that they can interact with the instantiated existentials.[12]

12 There are proof systems that sidestep this issue. See, for example Mates Elementary Logic1972 or Forbes Modern Logic 1994. These systems are all equivalent, but each has its own issues. Mates and Forbes do not use scope lines and so one loses the visual representation of logical dependencies.

Appendix Chapter 6

Derived rules:

QE
If you have a line ¬∀¬xφx, you may write ∃xφx on a subsequent line. Cite the original line containing ¬∀¬xφx, and the rule QE ("Quantifier Exchange")

.
.
7) ¬∀¬xPirate(x)
8) ∃xPirate(x) 7 QE
.
.

QE
If you have a line ∃xφx, you may write ¬∀¬xφx on a subsequent line. Cite the original line reading ∃xφx, and the rule QE ("Quantifier Exchange")

.
.
7) ∃xPirate(x)
8) ¬∀¬xPirate(x) 7 QE
.
.

QE
If you have a line ¬∃¬xφx, you may write ∀xφx on a subsequent line. Cite the original line containing ¬∃¬xφx, and the rule QE ("Quantifier Exchange")

.
.
7) ¬∃¬Pirate(x)
8) ∀xPirate(x) 7 QE
.
.

QE

If you have a line ∀xφx, you may write ¬∃¬xφx on a subsequent line. Cite the original line containing ∀xφx, and the rule QE ("Quantifier Exchange")

.
.
7) ∀xPirate(x)
8) ¬∃¬Pirate(x) 7 QE
.
.

Chapter 6 Appendix 2

The Belcher Pentapartite Strategery

(Note: '⊢′ means 'entails')

Phase 1: Work backwards.
Is your conclusion a conditional? If so, set up **→ Intro**: Assume the antecedent, the last line of the subproof is the consequent.

Is your conclusion a Universal? If so, set up a Universal intro by selecting an arbitrary object (never appeared in the proof before). The last line of the subproof will be the same as the universal you are trying to derive, only using your universal object instead of the universal quantifier and its bound variable.

Phase 2: Distribute negations through quantifiers
If you have any negated quantifiers, e.g., ¬∃xPirate(x) or ¬∀xPirate(x), use Quantifier Equivalence **(QE)** to distribute them:

¬ExPirate(x) ⊢ Ax¬Pirate(x)
¬AxPirate(x) ⊢ **¬**Ex¬Pirate(x)

Phase 3: Strip off Quantifiers
Do you have an Existential in the premises? If so, eliminate it with **Existential elim**. Pick a particular arbitrary object (never appeared in the proof before) and use it in place of the existentially bound variable. The last line of the subproof will be the same as the line you are trying to derive, but it can't have the arbitrary name in it anymore.
If you have universals, use **∀ Elim**: A**x Pirate(x) ⊢ Pirate(a)** *for any name a at all*! No subproofs, no arbitrary requirements, etc. Be careful: save ∀ Elim for as late as possible - always do ∃ elim and ∀ intro first!

Phase 4: Booleans
Break down all your Booleans, and build them back up, using
Break down:
DeMorgans, Disjunctive Syllogism, Conditional Equivalence, **MT**, → **Elim**, ∧ **Elim**, ↔ **Elim**, and so on.
Build up:
DeMorgans (remember it goes both ways), ∧ **Intro**, ∨ **Intro**, and so on.

Phase 5: The Hard Stuff

If you have disjunctions in the premises that you can't break up, try v elim. Make a separate subproof where you assume each disjunct, and try to derive the conclusion from each disjunct. Example:

1) Pirate(albert) ∨ Squirrel(albert)	
2) Pirate (a)	Assume
:	
:	
5) Plunders (albert, tortuga)	
6) Squirrel (albert)	Assume
:	
:	
9) Plunders (albert, tortuga)	
10) Plunders (albert, tortuga)	∨ Elim 1, 2-5, 6-9

If you're stumped, try Negation Intro. Assume the OPPOSITE of what you *want* to prove, then derive a contradiction. Then exit the assumption/ subproof and you can state the NEGATION of what you assumed. Example:

:	
:	
3) Vicious(albert)	Assume
:	
:	
4) ⊥	⊥ Intro (cite contradictory lines!)
5 ¬Vicious (albert)	¬ Intro / RAA 3-4

Chapter 7

Semantics and Translation for Relations and Multiple Quantifiers

It may seem a bit misleading to have a completely new section for this. After all, there are no new rules, no new principles, no new techniques. But the application of the old rules and techniques gets a bit tricky when there are multiple quantifiers, so I thought it would be best to start with the simpler sentences in chapter 2, and build on that here.

The paradigm cases we are looking at here are sentences involving relations, where both argument spots are quantified over by distinct quantifier phrases. For example, 'Someone pillages everything'. We'll go over the translation techniques in the next chapter; for now, bear with me that the correct translation is

$$\forall x(\text{Person}(x) \rightarrow \exists y\text{Pillages}(x,y))$$

That is, for all x, if x is a person, then there exists some y, and x pillages y. *The pillagee need not be the same person!*

Let's go through this somewhat informally ('How to read relational logic') and then deal with truth in a model.

In this sentence, 'everything' is playing the role of the subject. Well, not really, but close. The sentence is saying that everything has a conditional property: if they are a person, then something else follows. What follows? They pillage something. That is, any persons in the Domain $\mathscr{D}$ must stand in the pillaging relation to at least one thing. That is, the extension of the pillaging relation had better include a series of pairs <a, b>, where a in each pair is a person (until the people are exhausted) and b is anything whatsoever. Let's look at a model and see how this works.

$\mathcal{M}_1 = \langle \mathcal{D}_1, \mathcal{I}_1, \mathcal{V} \rangle$
$\mathcal{D}_1$ = {albert, wooster, gygax}
$\mathcal{I}_1$(Person) = {albert, gygax}
$\mathcal{I}_1$(Pillages) = {<albert, wooster> <wooster, wooster>}

You may think this model seems odd. Surely Wooster is a person. But I say, in this model, Wooster is a desk. Maybe it is a desk-fixated society and they name their desks. And Wooster pillages Wooster? Hey, stranger things have happened. There's probably an internet site devoted to self-pillaging. OK. Is

$\forall x(\text{Person}(x) \rightarrow \exists y\text{Pillage}(x,y))$

true in $\mathcal{M}_1$? The overall sentence has the form of a Universal (conditional). What is the truth rule for Universals? A sentence of the form

$\forall x \varphi x$

is true-in-$\mathcal{M}$ *iff* every element in the domain satisfies the open sentence φx. Here's where things get a little tricky, because φx is itself quantified. But as long as we are careful, this need not be a problem.

Luckily there are only three elements in the domain – Albert, Wooster, and Gygax. Do they each satisfy φx? First let's look at φx. It is the conditional,

$\text{Person}(x) \rightarrow \exists y\text{Pillages}(x,y)$.

Does Albert satisfy it? Looking at the extension of 'person', we see that both Albert and Gygax are persons. Thus, the antecedent of the conditional is true, and so the consequent must also be true in order for Albert and Gygax to satisfy the conditional. Again, since the consequent is itself quantified, that will take a little work. What about Wooster? Well, Wooster is not a person, so the antecedent is false, and if the antecedent is false, the conditional is true. Wooster satisfies the open sentence.

So now we know that for Albert or Gygax to satisfy the open sentence, they must satisfy the consequent $\exists y\text{Pillages}(x,y)$. That is an existential. How do you satisfy an existential? With strong black coffee and little cigarettes. Well, formally speaking, if there is one object in the domain that is pillaged by x. Is there an object in the domain that is pillaged by Albert? Yes – Wooster. We can tell this by looking at $\mathscr{I}_1$(Pillages) and seeing if there is a pair with the first member being Albert. There is: <albert. wooster> is in $\mathscr{M}_1$'s extension of the pillaging relation. What about Gygax? Does he pillage anything? Again, looking at $\mathscr{I}_1$(Pillages), we see the answer this time is no: there is no pair <gygax, anything> in the extension of pillaging.

So. Although Wooster pillages something, the other person in the domain, Gygax, doesn't pillage anything. So not everyone in $\mathscr{M}_1$ pillages something. So

$\forall x(\mathrm{Person}(x) \rightarrow \exists y \mathrm{Pillage}(x,y))$ is false in $\mathscr{M}_1$.

The process really works just the way it has since Boolean logic: evaluate a sentence by first evaluating the truth of its operator / quantifier with the widest scope, or the connective with the widest scope (the "central connective"). To do that, you may need to evaluate the truth of its component sentences, by evaluating *their* widest scope quantifier/operator/connective. And so on until one works down to the atomic sentences.

Just like the order of the connectives mattered, the order of the quantifiers matter. Recall that a negated disjunction has very different truth conditions from a disjunction of negations. So too does a sentence like

$\forall x \exists y Pxy$

differ from a sentence like

$\exists y \forall x Pxy$.

In the former, 'everything' is the pseudo-subject of the sentence (really, each instance of 'everyone' is the subject); In the latter, 'something' is the subject. The former says, everything has a property: there's something (not necessarily the same thing) that it pillages. The latter says, there's at least one thing that has a certain property: everything pillages it. (Who is this universally hated person? Justin Bieber?) Let's return to the endless drudgery of models to show this.

$\mathscr{M}_1$ = <$\mathscr{D}_1$, $\mathscr{I}_1$, $\mathscr{V}$>
$\mathscr{D}_1$ = {albert, wooster}
$\mathscr{I}_1$(Pillages) = {<albert, wooster> <wooster, albert> }

First consider ∀x∃yPillages(x,y). The universal has the widest scope, so we need to check if everything in the domain satisfies the open sentence ∃yPillages(x,y). I've kept the domain nice and small, so there's only two things to check – Albert and Wooster. Does Albert satisfy the open sentence

∃yPillages(x,y)[13]?

To determine that we need to evaluate the existential. For ∃yPillages(x,y) to be true in this case, at least one thing must satisfy the open sentence Pillages(albert, y). Yes – Wooster satisfies that. So Albert satisfies ∃yPillages(x,y). Okay, one thing in the domain satisfies ∃yPillages(x,y), but the overall claim is a universal. So everything in the domain needs to satisfy it. That leaves Wooster. So something better satisfy Pillages(wooster, y). In fact, a quick look at $\mathscr{I}_1$(Pillages) shows that Albert satisfies Pillages(wooster, y). So everything does satisfy ∃yPillages(x,y), and ∀x∃yPillages(x,y) is true in $\mathscr{M}_1$. Even if it is some sort of sick, perverse scenario of mutual pillaging.

What about ∃y∀xPillages(x,y)? Now we just have to find one element of the domain – Wooster or Albert, in this case – that satisfies the open sentence ∀xPillages(x,y). Is there anyone that everyone pillages? Let's start with everyone's favorite butler, Wooster. Does Wooster satisfy ∀xPillages(x,y)?

---

13 It's an open sentence, even though there is a quantifier phrase, because there's now that one unbound variable,

That's a universal, so Wooster satisfies it if every element in the universe – including Wooster! we did not say 'everything else', we said 'everything![14] - satisfies the open sentence Pillages(x, wooster). A quick look at $\mathscr{I}_1$(Pillages) shows that Wooster does not pillage himself. (He probably does, but he lies about it). That is, <wooster, wooster> is not in M1's extension of the pillaging relation. OK, so Wooster is not a guy everything pillages. But for ∃y∀xPillages(x,y) to be true in $\mathscr{M}_1$, just one individual must be all-pillaged. Maybe it's Albert. But the same reasoning we just used shows that Albert doesn't pillage himself, so he's not pillaged by everyone. In fact, no one is pillaged by everyone in this model, so ∃y∀xPillages(x,y) is false in $\mathscr{M}_1$. What did we just prove? (Take a second before turning).

14 And of course it's possible for Jeeves to pillage Jeeves. The British call that an "own-goal".

There's a model where $\forall x \exists y$Pillages(x,y) is true and $\exists y \forall x$Pillages(x,y) is false. Thus. . .? Right:

The two sentences are not logically equivalent.

The same follows for validity, consistency, etc. An argument is valid just in case every model that makes the premises true, makes the conclusion true; sentences are consistent just in case there is a model in which they are all true, a sentence is logically true just in case it is true in every model, and so on. The definitions for the logical concepts remain as they are.

Translation

Complex quantified noun phrases.

In QL with multiple quantifiers, the step by step method will be crucial. They are often too complicated to do all at once.

Sometimes a noun phrase itself as a quantifier in it. Just look for the presence of those quantifier words, or think about the definition of the word (e.g., a misanthrope is someone who hates everyone). Those are easy to translate once you get the knack. Consider

Everyone loves a pirate who pillages everything.

The pseudo-subject here is everyone, and the object of their love is *a pirate who loves everything.* Beware: 'a pirate' looks an awful lot like an existential doesn't it" But in English, the far more natural reading is 'any pirate'. You may have to think a bit about what is meant. Try passivizing and sticking the words 'in general' in front:

In general, a pirate who pillages everything is loved by all.
In general, a Guinness is just the thing for breakfast.

In each case, those sentences seem to make perfect sense, more so than:

There is a particular pirate who pillages everything, and everyone loves him.

There is a particular Guinness, and it is just the thing for breakfast.

So 'a pirate' and 'a Guinness' probably mean, 'any pirate' and 'any Guinness'. Now to translate the entire noun phrase:

a pirate who pillages everything.

Since we already said 'a' was 'any', this is about anyone who has each of two properties, being a pirate and pillaging everything:

$\forall$x(Pirate(x) $\wedge$ pillages-everything(x))

How to deal with the second conjunct? Try our old trick of passivizing to get the quantifier out front of the clause:

everything-is-pillaged-by(x)

Again, this looks like a straightforward quantifier, everything has some property (the property of being pillaged by x):

$\forall$yPillages(x, y)

Stop for a second to make sure x and y are in the right places: x is the pirate who is the pillager, and y is the thing pillaged. Legit. Now embed this clause in the earlier quantified sentence:

$\forall$x(Pirate(x) $\wedge$ **$\forall$yPillages(x, y)**)

(The bold type indicates what we just embedded).

Now, on its own, this is an extremely strong claim, the claim that everything is a pirate who pillages everything. Except in very strange models, this is probably false. One would almost never make this claim on its own. It will really only in practice be used when embedded inside a larger sentence, say a conditional like the one we are considering. Let's hold off on that for a bit, though.

Translating: The Step-By-Step Method

Now back to our original sentence,

Everyone loves a pirate who pillages everything.

That's an Aristotelean form, All Fs are G, or

∀x(Person(x) → Loves(x, **any Pirate who pillages everything**))

Again the bold indicates what we just embedded. Careful! We already know that our embedded quantified noun phrase has bound variables x and y in it, so we'd better change that outermost variable:

∀z(Person(z) → Loves(z, **any Pirate who pillages everything**))

In our embedded noun phrase, let's again passivize to get that quantifier out front:

any pirate who pillages everything is loved by z

Looks like All Fs are Gs, again:

∀x(Fx → Gx)

with F being 'any pirate who pillages everything', and G being 'loved by z'.We already did F, so

∀x(**(Pirate(x) ∧ ∀yPillages(x, y))** → Loves(z, x))

Stop for a moment to check that the variables are in the right places. Also, note that while

∀x(Pirate(x) ∧ ∀yPillages(x, y))

was far too strong all on its own, as the antecedent of a conditional the claim is much weaker. It's no longer saying that everything is an omnipillaging pirate, but rather, that everything that *is* an omnipillaging pirate has some other property (being universally loved, in this case). Let's embed this in the larger sentence now:

$\forall z(\text{Person}(z) \rightarrow \forall x((\textbf{Pirate(x)} \wedge \forall \textbf{yPillages(x, y))} \rightarrow \textbf{Loves(z, x)}))$

Everything z, if it's a person, has this property: for all omnipillaging pirates, z loves them.

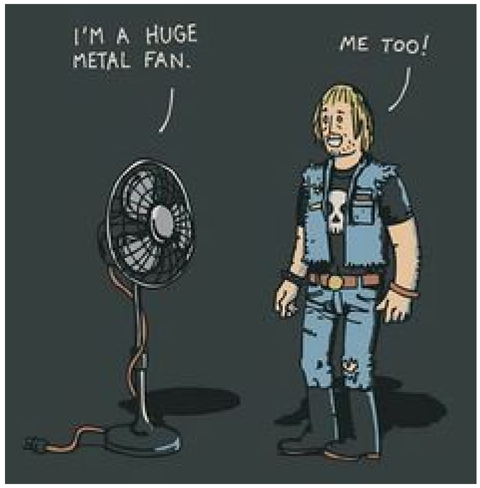
Notice that the sentence

$\forall z\forall x\forall y(\text{Person}(z) \rightarrow ((\textbf{Pirate(x)} \wedge \textbf{Pillages(x, y))} \rightarrow \textbf{Loves(z, x)}))$

Is equivalent. But I think that sort of translation is bad practice. It hides the fact that 'a pirate who pillages everything' is itself a complex (quantified) noun phrase, a syntactic and semantic structure in its own right. It "spreads" the structure of the NL sentence over the whole sentence. I think best practices are to try to preserve as much of the NL surface structure as transparently as possible. This often means giving quantifier phrases the narrowest scope the could have (without creating unbound variables).

Ambiguity

Ambuguity comes from two sources: lexical ambiguity, and syntactic ambiguity. Lexical ambiguity is when a word means two (or more) things, like 'bank': a place that steals your money, and the edge of a river. Sometimes phrases can be ambiguous this way:



(Although, that's probably simply the result of a double lexical ambiguity, in 'fan' and 'metal'). PL and by extension QL rule out lexical ambiguity by insisting that every word have just one referent or meaning. When you translate you must be aware of multiple possible meanings.

Syntactic ambiguity, on the other hand, can occur even when you hold the meanings of the individual words stable. Examples include :

> This morning I shot an elephant in my pajamas. How he got in my pajamas I don't know. (Groucho Marx)
> The professor said on Monday he would give an exam.
> Miners refuse to work after death.

In each case here, the ambiguity arises from the sentence structures that are compatible. For example, does 'in my pajamas' apply to the shooter, or the elephant? Again, PL and QL rule out such ambiguities by making sentence structure explicit; in effect, this correlates every sentence with exactly one meaning. But in translating, you have to determine which structure the NL sentence has – sometimes an impossible task, if you don't have enough context! Let's look at a classic:

> Every girl loves some boy

This has two possible meanings:

> Every girl has a property: there's some boy – not necessarily the same one – that they love.
> There's a boy, and he's got a property: every girl loves him.

Which is being expressed by the sentence? It is impossible to tell. Maybe if you were talking with your buddy Wooster, and he had been telling you about being at the grocery store and seeing the People Magazine's “Sexiest Man of the Year” issue, you would take him to mean the latter (there is a boy beloved of all girls). And maybe if you were talking to a member of the Westboro Baptist Church, and they were stridently denying the existence of lesbians, you might take them to mean the former. It's our job as logicians to show the different QL sentence structures of these sentences.

Interpretation 1: Every girls has a boy they love.

Well, that sure looks like, all Fs are Gs. So the Aristotelean skeleton of the sentence is:

∀x(G(x) → Loves-some-boy(x))

There's a quantifier in the consequent, so let's try our passivization trick to see if we can get the quantifier out front and perhaps reveal a hidden Aristotelean structure:

∀x(G(x) → some-boy-is-loved-by(x))

Sure enough we've revealed an Aristotelean form: Some Fs (boys) are loved by x.

∀x(G(x) → **∃y(B(y) ∧ loved-by(y, x)**)

Notice: We passivized 'loves' to 'is loved by', so the subject/object (given by the argument places) is switched. It is, after all, y, the boy, that has the property of being loved by all girls. But just like your sixth grade grammar teacher told you, we don't like to use passive voice, so change 'loved by' back to loves:

∀x(G(x) → **∃y(B(y) ∧ loves(x, y)**)

And just as before, change your argument places to make sure that the boy is loved, and the girl the lover. Argument space 1 is the lover – yes, that's x, the girl – and argument space 2 is the beloved – yes that's the boy.

Interpretation 2: There's at least one boy, and he's special - every girl loves him.

Looks like a clear cut case of the Aristotelean form Some Fs are Gs:

$\exists x(B(x) \wedge \text{all girls love}(x))$

The consequent has a quantifier in it, but we don't need to pull any cute tricks to see that it too is an Aristotelean form: All Fs (girls) are G(s) (lovers of (x)))

$\exists x(B(x) \wedge Ay(Girl(y) \rightarrow Loves(y, x)))$

As before, we check to make sure the xs and ys are in the right place. Yup.

Chapter 8

Proofs Involving Multiple Quantifiers and Relations

In a sense, there really isn't any reason for this chapter – there's nothing new in principle, just added layers of complexity. The only new issue is the need to think about what names you will instantiate your quantified sentences with. Sometimes you may need to think ahead a few moves for this. Otherwise, prrofs go the same way – just use the pentapartite strategery.

So instead of going over the rulesw and the strategery, which we've already done, here we will walk through some proofs with multiple, overlapping quantifiers. Let's start with that old chestnut,

Every girl loves some boy

As we saw, this has two possible interpretations:

There's a boy that every girl loves, and
Every girl has a (potentially distinct) boy that they love.

We saw that these are not equivalent. But does one entail the other, perhaps? I will cut to the chase: if there is a single boy beloved of all girls, then it must be that every girl has a boy (that one) that they love. It does not work the other direction: If every girl loves a possibly distinct boy, it does not follow that there's a single boy they all love. So:

$\exists x(Boy(x) \land \forall y(Girl(y) \rightarrow Loves(y, x))) \models \forall x(Girl(x) \rightarrow \exists x(B(x) \land Loves (x,y)))$

Let us prove it.

1) $\exists x(Boy(x) \land \forall y(Girl(y) \rightarrow Loves(y, x)))$
.
.
.
.
BB) $\forall x(Girl(x) \rightarrow \exists y(Boy(y) \land Loves (x,y)))$

('BB' is 'Blackbeard'; the theme is pirates.)

There's no conditionals in the conclusion (well, one is embedded, but that doesn't count). But the conclusion is a universal, so it is natural to start off with a ∀intro. Keep in mind that it's often rather flexible whether you build up your universals first (Phase 1) or strip off your existentials first (Phase 3). We could do either here, but let's start with the universal:

1) ∃x(Boy(x) ∧ ∀y(Girl(y) → Loves(y, x)))	
2) [hanna]	
.	
.	
.	
JS) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y)))	?
BB) ∀x(Girl(x) → ∃y(Boy(y) ∧ Loves (x,y)))	∀intro 2-JS

Albert is our universal object. Ha he been mentioned before 2? No. So the only way we could prove anything about Albert is if we could prove it about anything in general. Hence, if we can get to JS (Jack Sparrow), we can universally generalize over Albert in BB.

1) $\exists x(Boy(x) \wedge \forall y(Girl(y) \rightarrow Loves(y, x)))$

2)
hanna

3)
albert

4) $Boy(albert) \wedge \forall y(Girl(y) \rightarrow Loves(y, albert))$

.

.

MC) $(Girl(hanna) \rightarrow \exists y(Boy(y) \wedge Loves(hanna,y)))$?

JS) $(Girl(hanna) \rightarrow \exists y(Boy(y) \wedge Loves (hanna,y)))$ ∃elim 1, 3-MC

BB) $\forall x(Girl(x) \rightarrow \exists y(Boy(y) \wedge Loves (x,y)))$ ∀intro 2-JS

(4) is just the existential elim of line 1, in favor of Albert. But we don't get to cite ∃elim until the subproof is over, and the name 'Albert' has been shown arbitrary and temporary. Well, by (MC) (Madame Cheng), Albert is out of the picture, and he never appeared anywhere before. So our citing ∃elim in (JS) is legit. Let's bust up that conjunction in (4), now:

1) ∃x(Boy(x) ∧ ∀y(Girl(y) → Loves(y, x)))

2)
hanna

3)
albert

4) Boy(albert) ∧ ∀y(Girl(y) → Loves(y, albert))

5) Boy(albert) ∧elim 4

6) ∀y(Girl(y) → Loves(y, albert)) ∧elim 4

.
.
.

MC) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) ?

JS) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) ∃elim 1, 3-MC

BB) ∀x(Girl(x) → ∃y(Boy(y) ∧ Loves (x,y))) ∀intro 2-JS

Looks like our only option now is to set up a conditional intro for (MC).

1) ∃x(Boy(x) ∧ ∀y(Girl(y) → Loves(y, x)))

2)
hanna

3)
albert

4) Boy(albert) ∧ ∀y(Girl(y) → Loves(y, albert))

5) Boy(albert) ∧elim 4

6) ∀y(Girl(y) → Loves(y, albert)) ∧elim 4

7) Girl(hanna)

.

.

CK) ∃y(Boy(y) ∧ Loves (hanna,y)) ?

MC) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) →intro 7-CK

JS) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) ∃elim 1, 3-MC

BB) ∀x(Girl(x) → ∃y(Boy(y) ∧ Loves (x,y))) ∀intro 2-JS

(CK) (Captain Kidd)is a conjunction[15], so we will probably need to prove each conjunct separately.

What line is even left to break down? Looks like (6) is all that is left.

15 That's a lie. It's not a conjunction. It's an existential. But the embedded sentence is a conjunction, so if we prove the conjunction, we can just ∃intro over it.

1) ∃x(Boy(x) ∧ ∀y(Girl(y) → Loves(y, x)))

2)
hanna

3)
albert

4) Boy(albert) ∧ ∀y(Girl(y) → Loves(y, albert))

5) Boy(albert) ∧elim 4

6) ∀y(Girl(y) → Loves(y, albert)) ∧elim 4

7) Girl(hanna) Assume

8) (Girl(hanna) → Loves(hanna, albert)) ∀elim 6

.

.

CK) ∃y(Boy(y) ∧ Loves (hanna,y)) ?

MC) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) →intro 7-CK

JS) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) ∃elim 1, 3-MC

BB) ∀x(Girl(x) → ∃y(Boy(y) ∧ Loves (x,y))) ∀intro 2-JS

It ought to be easy-peasy propositional logic (Boolean) from here, with the exception of (CK), which will probably (but not certainly) be an ∃intro:

1) $\exists x(\text{Boy}(x) \wedge \forall y(\text{Girl}(y) \rightarrow \text{Loves}(y, x)))$

2)
hanna

3)
albert

4) $\text{Boy}(\text{albert}) \wedge \forall y(\text{Girl}(y) \rightarrow \text{Loves}(y, \text{albert}))$

5) Boy(albert) — $\wedge$elim 4

6) $\forall y(\text{Girl}(y) \rightarrow \text{Loves}(y, \text{albert}))$ — $\wedge$elim 4

7) Girl(hanna) — Assume

8) $(\text{Girl}(\text{hanna}) \rightarrow \text{Loves}(\text{hanna}, \text{albert}))$ — $\forall$elim 6

9) Loves(hanna, albert) — $\rightarrow$elim 7,

.

.

CK) $\exists y(\text{Boy}(y) \wedge \text{Loves}(\text{hanna},y))$ — ?

MC) $(\text{Girl}(\text{hanna}) \rightarrow \exists y(\text{Boy}(y) \wedge \text{Loves}(\text{hanna},y)))$ — $\rightarrow$intro 7-CK

JS) $(\text{Girl}(\text{hanna}) \rightarrow \exists y(\text{Boy}(y) \wedge \text{Loves}(\text{hanna},y)))$ — $\exists$elim 1, 3-MC

BB) $\forall x(\text{Girl}(x) \rightarrow \exists y(\text{Boy}(y) \wedge \text{Loves}(x,y)))$ — $\forall$intro 2-JS

Look at our goal, CK. It's an existential. what would it look like if 'y' was replaced by some particular? after all, something has to satisfy the unquantified part of CK, viz.,

$\text{Boy}(y) \wedge \text{Loves}(\text{hanna},y)$

in order for CK to follow. Just one stinking thing. Do we know of any boys? Yes, Albert. So is Albert loved by Hanna? As it turns out, yes – see (9). So maybe our new subgoal is:

$\text{Boy}(\text{albert}) \wedge \text{Loves}(\text{hanna},\text{albert})$

1) ∃x(Boy(x) ∧ ∀y(Girl(y) → Loves(y, x)))

2)
hanna

3)
albert

4) Boy(albert) ∧ ∀y(Girl(y) → Loves(y, albert))

5) Boy(albert) ∧elim 4

6) ∀y(Girl(y) → Loves(y, albert)) ∧elim 4

7) Girl(hanna) Assume

8) (Girl(hanna) → Loves(hanna, albert)) ∀elim 6

9) Loves(hanna, albert) →elim 7,

.

SK) Boy(albert) ∧ Loves (hanna,albert) ?

CK) ∃y(Boy(y) ∧ Loves (hanna,y)) ∃intro

MC) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) →intro 7-CK

JS) (Girl(hanna) → ∃y(Boy(y) ∧ Loves (hanna,y))) ∃elim 1, 3-MC

BB) ∀x(Girl(x) → ∃y(Boy(y) ∧ Loves (x,y))) ∀intro 2-JS

(SK) is for Shirahama Kenki, a 16-17th century pirate. Who said Logic wasn't diverse? Anyway the move from SK to CK is just replacing 'Albert' with 'some y'. Now we just need each conjunct in (SK). But we've broken everything down already. Do we have them? Yes, (5) and (9). Sweet.

1) $\exists x(\text{Boy}(x) \land \forall y(\text{Girl}(y) \to \text{Loves}(y, x)))$	
2) hanna	
3) albert	
4) $\text{Boy}(\text{albert}) \land \forall y(\text{Girl}(y) \to \text{Loves}(y, \text{albert}))$	
5) $\text{Boy}(\text{albert})$	$\land$elim 4
6) $\forall y(\text{Girl}(y) \to \text{Loves}(y, \text{albert}))$	$\land$elim 4
7) $\text{Girl}(\text{hanna})$	Assume
8) $(\text{Girl}(\text{hanna}) \to \text{Loves}(\text{hanna}, \text{albert}))$	$\forall$elim 6
9) $\text{Loves}(\text{hanna}, \text{albert})$	$\to$elim 7,
10) $\text{Boy}(\text{albert}) \land \text{Loves}(\text{hanna},\text{albert})$	$\land$Intro 5,9
11) $\exists y(\text{Boy}(y) \land \text{Loves}(\text{hanna},y))$	$\exists$intro 10
12) $(\text{Girl}(\text{hanna}) \to \exists y(\text{Boy}(y) \land \text{Loves}(\text{hanna},y)))$	$\to$intro 7-11
13) $(\text{Girl}(\text{hanna}) \to \exists y(\text{Boy}(y) \land \text{Loves}(\text{hanna},y)))$	$\exists$elim 1, 3-12
BB) $\forall x(\text{Girl}(x) \to \exists y(\text{Boy}(y) \land \text{Loves}(x,y)))$	$\forall$intro 2-13

Done! Notice I have cleaned up the line numbers.

Let's try one last one, putting everything together – translation and proof. This one is very hard.

Everyone loves a lover
Someone loves someone
∴ Everyone loves everyone

Sounds like that conclusion is quite a bit to prove, eh? Well, on at least one reading – the first sentence is ambiguous in a few ways! – the argument is in fact valid.

First, lets look at the first sentence, everyone loves a lover. This is clearly All Fs are G, so our "skeleton" of the sentence is:

$\forall$x(Person(x) $\rightarrow$ x loves a lover)

Now we have a choice to make. Does that mean that x loves a particular lover, i.e., for every person in the domain, there's a lover that they love? That would net us:

$\forall$x(Person(x) $\rightarrow$ x loves a lover)

or

$\forall$x(Person(x) $\rightarrow$ x loves some lover)

and passivizing:

$\forall$x(Person(x) $\rightarrow$ some lover is loved by x)

So the consequent there is also an Aristotelean form, namely, some Fs are G:

$\forall x(\text{Person}(x) \rightarrow \exists y(\text{Lover}(y) \wedge \text{Loves}(x, y))$

Now, the predicate 'Lover' is itself a complex quantified noun. But we'll deal with that in a bit. On this interpretation, I am pretty sure the argument is invalid. But there's another way to read the first sentence. It might mean that lovers are the sorts of people that everyone loves, so 'a lover' would be treated as 'in general, people love (all) lovers)'. Sort of like 'a shot of whiskey is just the thing for breakfast.' In that case, sentence one would read:

$\forall$x(Person(x) $\rightarrow$ lovers are loved by x)

or

$\forall$x(Person(x) $\rightarrow$ all lovers are loved by x)

and now the consequent is another Aristotelean form, all Fs are G:

$\forall$x(Person(x) $\rightarrow$ $\forall$y(lover(y) $\rightarrow$ Loves(x,y))

And now we have yet another choice. recall I said that 'Lover' was itself a complex quantified noun. What is a Lover? Is it someone who loves at least one person, or is it someone who loves everyone? I suppose it could go either way, but I think that if lovers were people that loved everyone, there wouldn't be any lovers. Also, it seems to me that (say) Romeo was a lover, but he only loved one person (Juliet, right? Maybe he loved his mom. But not everyone. He hated Tybalt, for sure.) So if that's right, 'y is a lover' is

Ez(Person(z) $\wedge$ Loves(y, z))

Notice that there is a free variable there, namely y. That's alright – we want a predicate, and a predicate has free variables. It'll be bound in the larger sentence when we replace Lover(y) with Ez(Person(z) $\wedge$ Loves(y, z)):

$\forall$x(Person(x) $\rightarrow$ $\forall$y($\exists$z(Person(z) $\wedge$ Loves(y, z)) $\rightarrow$ Loves(x,y))

In (tortured) English:

Every person (x) has this property: for every person (y) (including themselves), if that person loves someone (including themselves) (z), the first person (x) loves that person (z). Phew!

Premise 2 – someone loves someone – and the conclusion – everyone loves everyone – are pretty straightforward.

Someone loves someone

Is clearly an Aristotelean form, Some Fs are G:

$\exists x(\text{Person}(x) \wedge \text{Loves someone}(x))$

Passivizing:

$\exists x(\text{Person}(x) \wedge \text{someone is loved by } x)$

The second conjunct, again, is an Aristotelean form just like the first, Some Fs are G:

$\exists x(\text{Person}(x) \wedge \exists y(\text{Person}(y) \wedge \text{Loves}(x, y))$

(notice that x goes back into the first argument slot, since we are returning to active voice, and it's x who is doing the loving).

The conclusion is practically the same, only with universals.

everyone loves everyone

Is another Aristotelean form, All Fs are G:

$\forall x(\text{Person}(x) \rightarrow x \text{ loves everyone})$

Passivizing once more we get:

∀x(Person(x) → everyone is loved by x)

And once again (surprise!) Our antecedent is an Aristotelian form, All Fs are G:

∀x(Person(x) → Ay(Person(y) → Loves (x,y)))

To sum up: Our argument, formalized, is:

1) ∀x(Person(x) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(x,y))
2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))
3) ∴ ∀x(Person(x) → Ay(Person(y) → Loves (x,y)))

That was the easy part! Let's set up the proof:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y))	Premise
2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))	Premise
.	
.	
.	
3) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	?

First off, looking at the strategery, we see we want to get rid of that existential in line 2. So we set up an E elim:

1) $\forall x(\text{Person}(x) \rightarrow \forall y(\exists z(\text{Person}(z) \wedge \text{Loves}(y, z)) \rightarrow \text{Loves}(x,y))$	Premise
2) $\exists x(\text{Person}(x) \wedge \exists y(\text{Person}(y) \wedge \text{Loves}(x, y)))$	Premise
3)	
a	
$(\text{Person}(a) \wedge \exists y(\text{Person}(y) \wedge \text{Loves}(a, y))$	Assume
.	
.	
jp) $\forall x(\text{Person}(x) \rightarrow \forall y(\text{Person}(y) \rightarrow \text{Loves }(x,y)))$	?
Conc) $\therefore \forall x(\text{Person}(x) \rightarrow \forall y(\text{Person}(y) \rightarrow \text{Loves }(x,y)))$	$\exists$ elim 2, 3-jp

(jp) is for 'Judas Priest). Next we do the obvious and break up the conjunction in line 3:

1) $\forall x(\text{Person}(x) \rightarrow \forall y(\text{Ez}(\text{Person}(z) \wedge \text{Loves}(y, z)) \rightarrow \text{Loves}(x,y))$	Premise
2) $\exists x(\text{Person}(x) \wedge \text{Ey}(\text{Person}(y) \wedge \text{Loves}(x, y)))$	Premise
3)	
a	
$\text{Person}(a) \wedge \exists y(\text{Person}(y) \wedge \text{Loves}(a, y))$	Assume
4) $\text{Person}(a)$	$\wedge$ elim 3
5) $\exists y(\text{Person}(y) \wedge \text{Loves}(a, y))$	$\wedge$ elim 3
.	
.	
jp) $\forall x(\text{Person}(x) \rightarrow \forall y(\text{Person}(y) \rightarrow \text{Loves }(x,y)))$	?
Conc) $\therefore \forall x(\text{Person}(x) \rightarrow \forall y(\text{Person}(y) \rightarrow \text{Loves }(x,y)))$	$\exists$ elim 2, 3-jp

Strictly by the book so far. Damn, we have another existential to get rid of! Well, remember what I said about doing E elim and A intro before A elim. If you see and embedded existential, break up its embedding sentence to get at it first, just the way we did. Nothing for it but to set up another E elim and break up the resulting conjunction:

1) $\forall x(\text{Person}(x) \rightarrow \forall y(\text{Ez}(\text{Person}(z) \wedge \text{Loves}(y, z)) \rightarrow \text{Loves}(x,y))$	Premise
2) $\exists x(\text{Person}(x) \wedge \text{Ey}(\text{Person}(y) \wedge \text{Loves}(x, y)))$	Premise

3)
a
Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume
4) Person(a) ∧ elim 3
5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)
b
Person(b) ∧ Loves(a, b)
Assume
7) Person(b) ∧ elim 6
8) Loves(a, b) ∧ elim 6
.
.
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ?

jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim 5, 6-bs

Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim 2, 3-jp

('bs' is 'Black Sabbath'). It's starting to get hairy! Now it's time to get that last initial subproof, the universal intro in for the subconclusion, bs:

1) ∀x(Person(x) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise
2) ∃x(Person(x) ∧ Ey(Person(y) ∧ Loves(x, y))) Premise

3)

a

Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume

4) Person(a) ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)

b

Person(b) ∧ Loves(a, b) Assume

7) Person(b) ∧ elim 6

8) Loves(a, b) ∧ elim 6

9)

c

.

.

dp) Person(c) → ∀y(Person(y) → Loves (c,y))) ?

bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∀ in intro 9-dp

jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim 5, 6-bs

Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim 2, 3-jp

('dp' is 'deep purple'). Stop for a second and make sure the flagged names are all arbitrary and temporary. Yup, they are, so we are golden so far.

Now there are. . . choices. It will be tricky to instantiate that first universal. I mean, we can instantiate it any way we want; it's a universal, so that part is easy. But it will be difficult to instantiate it just right to make the a's, b's and c's all line up. Let's try it with 'c', based on where we want to go in line dp:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y))	Premise
2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y)))	Premise
3) a (Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y))	Assume
4) Person(a)	∧ elim 3
5) ∃y(Person(y) ∧ Loves(a, y))	∧ elim 3
6) b Person(b) ∧ Loves(a,b)	Assume
7) Person(b)	∧ elim 6
8) Loves(a,b)	∧ elim 6
9) c	
10) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,y)	∀ Elim 1
.	
.	
dp) (Person(c) → ∀y(Person(y) → Loves (c,y)))	?
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∀ intro 9-dp
jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 5, 6-bs
Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 2, 3-jp

Looks good so far. But wait – we only have a and b as persons, so we cannot do the required → elim on line 10!

Let's try again, with b instead, following up with another ∀elim:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y))	Premise
2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y)))	Premise
3) [a] (Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y))	Assume
4) Person(a)	∧ elim 3
5) ∃y(Person(y) ∧ Loves(a, y))	∧ elim 3
6) [b] Person(b) ∧ Loves(a,b)	Assume
7) Person(b)	∧ elim 6
8) Loves(a,b)	∧ elim 6
9) [c]	
10) Person(b) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y)	∀ Elim 1
11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y)	→elim 7, 10
12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c)	∀ Elim 11
.	
.	
dp) (Person(c) → ∀y(Person(y) → Loves (c,y)))	?
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∀ intro 9-dp
jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 5, 6-bs
Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 2, 3-jp

Hmmm. This doesn't appear to be helping. 'c' is just in the wrong place. No matter how we instantiate the x in line 1, the resulting line 11 will just have c in the wrong place(s).

Can we get crafty with our universals? We're going to have to. But first, let's set up the other parts of the proof that we are forced to do early – viz., our E elims and A intros.

We can start by setting an → intro up to get the conditional at (dp), namely, subproof 12-iron Maiden (im). Then, since (im) is a universal, we set up the ∀ intro at 13-slayer (sl).

1) $\forall x(Person(x) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(x,y))$ — Premise

2) $\exists x(Person(x) \wedge \exists y(Person(y) \wedge Loves(x, y)))$ — Premise

> 3) [a] $(Person(a) \wedge \exists y(Person(y) \wedge Loves(a, y))$ — Assume
>
> 4) $Person(a)$ — $\wedge$ elim 3
>
> 5) $\exists y(Person(y) \wedge Loves(a, y))$ — $\wedge$ elim 3
>
> > 6) [b] $Person(b) \wedge Loves(a,b)$ — Assume
> >
> > 7) $Person(b)$ — $\wedge$ elim 6
> >
> > 8) $Loves(a,b)$ — $\wedge$ elim 6
> >
> > > 9) [c]
> > >
> > > 10) $Person(b) \rightarrow \forall y(Ez(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$ — $\forall$ Elim 1
> > >
> > > 11) $\forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$ — $\rightarrow$elim 7, 10
> > >
> > > 12) $(\exists z(Person(z) \wedge Loves(c, z)) \rightarrow Loves(b,c)$ — $\forall$ Elim 11
> > >
> > > > 12) $Person(c)$ — Assume
> > > >
> > > > > 13) [e]
> > > > >
> > > > > .
> > > > >
> > > > > .
> > > > >
> > > > > sl) $Person(e) \rightarrow Loves (c,e)$ — ?
> > > >
> > > > im) $\forall y(Person(y) \rightarrow Loves (c,y))$ — $\forall$ Intro, 13-sl
> > >
> > > dp) $(Person(c) \rightarrow \forall y(Person(y) \rightarrow Loves (c,y)))$ — $\forall$ Intro, 12-im
> >
> > bs) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ — $\forall$ intro 9-dp
>
> jp) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ — $\exists$ elim, 5, 6-bs

Conc) $\therefore \forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ — $\exists$ elim, 2, 3-jp

(sl) is itself a conditional, so yet ANOTHER subproof and $\rightarrow$Intro (uh oh!):

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

3)
b
(Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume

4) Person(a) ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)
b
Person(b) ∧ Loves(a,b) Assume

7) Person(b) ∧ elim 6

8) Loves(a,b) ∧ elim 6

9)
c

10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1

11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10

12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11

12) Person(c) Assume

13)
e

14) Person(e) Assume
.
.
me) Loves(c,e) ?

sl) Person(e) → Loves (c,e) → Intro 14-me

im)∀y(Person(y) → Loves (c,y)) ∀ Intro, 13-sl

dp) (Person(c) → ∀y(Person(y) → Loves (c,y))) ∀ Intro, 12-im

bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∀ intro 9-dp

jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 5, 6-bs

Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 2, 3-jp (me) is Metallica, incidentally. Now we have to seriously thinkalize. Our goal is Loves(c,e). Do we see that sort of info in our premises anywhere? No. The only loving relation we have is L(a,b). Do we have anything like it, but with universally quantified variables? YES! Line 1 ultimately ends in the consequent L(x,y), where both x and y are universally quantified. So they can be c and e respectively, if we want. Let's try working backwards, then:

1) $\forall x(Person(x) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(x,y))$ Premise

2) $\exists x(Person(x) \wedge \exists y(Person(y) \wedge Loves(x, y)))$ Premise

3) a

$(Person(a) \wedge \exists y(Person(y) \wedge Loves(a, y))$ Assume

4) $Person(a)$ $\wedge$ elim 3

5) $\exists y(Person(y) \wedge Loves(a, y))$ $\wedge$ elim 3

6) b

$Person(b) \wedge Loves(a,b)$ Assume

7) $Person(b)$ $\wedge$ elim 6

8) $Loves(a,b)$ $\wedge$ elim 6

9) c

10) $Person(b) \rightarrow \forall y(Ez(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$ $\forall$ Elim 1

11) $\forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$ $\rightarrow$elim 7, 10

12) $(\exists z(Person(z) \wedge Loves(c, z)) \rightarrow Loves(b,c)$ $\forall$ Elim 11

12) $Person(c)$ Assume

13) e

14) $Person(e)$ Assume

15) $Person(c) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(c,e))$

$\forall$ elim 1

.

.

me) $Loves(c,e)$?

sl) $Person(e) \rightarrow Loves (c,e)$ $\rightarrow$ Intro 14-me

im)$\forall y(Person(y) \rightarrow Loves (c,y))$ $\forall$ Intro, 13-sl

dp) $(Person(c) \rightarrow \forall y(Person(y) \rightarrow Loves (c,y)))$ $\forall$ Intro, 12-im

bs) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ $\forall$ intro 9-dp

jp) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ $\exists$ elim, 5, 6-bs

Conc) ∴ $\forall x(\text{Person}(x) \rightarrow \forall y(\text{Person}(y) \rightarrow \text{Loves}(x,y)))$	$\exists$ elim, 2, 3-jp

In 15, we substitute c for the x in line 1, since we know we ultimately want c to be our lover. Luckily, line 12 tells us Person(c), so we can do an → elim on (mf), and then think: who do we want to be the loved, or y? We want e to be the loved. So:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise
2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

3)
a
(Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume
4) Person(a) ∧ elim 3
5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)
b
Person(b) ∧ Loves(a,b) Assume
7) Person(b) ∧ elim 6
8) Loves(a,b) ∧ elim 6

9)
c
10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1
11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10
12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11

12) Person(c) Assume

13)
e

14) Person(e) Assume
15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) ∀ elim 1
16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) → elim 12,15
17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e)) ∀ elim 16
me) Loves(c,e) ?

sl) Person(e) → Loves (c,e) → Intro 14-me
im)∀y(Person(y) → Loves (c,y)) ∀ Intro, 13-sl
dp) (Person(c) → ∀y(Person(y) → Loves (c,y))) ∀ Intro, 12-im
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∀ intro 9-dp

jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 5, 6-bs
Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 2, 3-jp

Okay, so now all that remains is to show that e loves some person – in PL,

∃z(Person(z) ∧ Loves(e, z))

then we can do a → elim on 16 for our answer. That shouldn't be too hard! Let's scribble in our new subgoal at fw (Fates Warning):

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

3)

a

(Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume

4) Person(a) ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)

b

Person(b) ∧ Loves(a,b) Assume

7) Person(b) ∧ elim 6

8) Loves(a,b) ∧ elim 6

9)

c

10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1

11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10

12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11

12) Person(c) Assume

13)

e

14) Person(e) Assume

15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e))

∀ elim 1

16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e))

→ elim 12,15

17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e))

∀ elim 16

.

.

fw) ∃z(Person(z) ∧ Loves(e, z)) ?

me) Loves(c,e) → elim 17, fw

sl) Person(e) → Loves (c,e) → Intro 14-me

im)∀y(Person(y) → Loves (c,y)) ∀ Intro, 13-sl

dp) (Person(c) → ∀y(Person(y) → Loves (c,y)))	∀ Intro, 12-im
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∀ intro 9-dp
jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 5, 6-bs
Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 2, 3-jp

Okay. Do we know that e loves anyone? And that the loved thing is a person? And e is a person? Line 13 tells us e is a person. Maybe we can make use of that double universally quantified Loves(x,y) at the tail of that giant conditional in 1, again, to show that e loves someone:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

3)

a

(Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume

4) Person(a) ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)

b

Person(b) ∧ Loves(a,b) Assume

7) Person(b) ∧ elim 6

8) Loves(a,b) ∧ elim 6

9)

c

10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1

11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10

12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11

12) Person(c) Assume

13)

e

14) Person(e) Assume

15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e))

∀ elim 1

16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e))

→ elim 12,15

17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e))

∀ elim 16

18) Person(e) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y))

∀ elim 1

.	
fw) ∃z(Person(z) ∧ Loves(e, z))	?
me) Loves(c,e)	→ elim 17, fw
sl) Person(e) → Loves (c,e)	→ Intro 14-me
im)∀y(Person(y) → Loves (c,y))	∀ Intro, 13-sl
dp) (Person(c) → ∀y(Person(y) → Loves (c,y)))	∀ Intro, 12-im
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∀ intro 9-dp
jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 5, 6-bs
Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 2, 3-jp

That's the hard part, right there – realizing that you have to instantiate line 1 two separate times. That's why QL proofs can never really be fully mechanized (if the domain is infinite), because you have to be clever about which instances of some quantifiers you want to use. (by the way, (fw) is Fates Warning).

Now a quick → elim of 18:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

3)
a
(Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume

4) Person(a) ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)
b
Person(b) ∧ Loves(a,b) Assume

7) Person(b) ∧ elim 6

8) Loves(a,b) ∧ elim 6

9)
c

10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1

11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10

12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11

12) Person(c) Assume

13)
e

14) Person(e) Assume

15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e))
∀ elim 1

16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e))
→ elim 12,15

17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e))
∀ elim 16

18) Person(e) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y))
∀ elim 1

19) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y)	
	→ elim 14, 18
.	
.	
fw) ∃z(Person(z) ∧ Loves(e, z))	?
me) Loves(c,e)	→ elim 17, fw
sl) Person(e) → Loves (c,e)	→ Intro 14-me
im)∀y(Person(y) → Loves (c,y))	∀ Intro, 13-sl
dp) (Person(c) → ∀y(Person(y) → Loves (c,y)))	∀ Intro, 12-im
bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∀ intro 9-dp
jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 5, 6-bs
Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y)))	∃ elim, 2, 3-jp

So close! Now we just need to show that anyone (since y is universally quantified in line 19) is both loved by e and is loved by some other person. Hmmm. Tricky. Look closely at line 19: The person loved by e ought to be the same person who loves someone. Do we already know a person who loves someone? Yes! a is a person (line 4) and a loves someone (viz, b, in line 8). So let's instantiate line 19's y for a:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

> 3) a
>
> (Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume
>
> 4) Person(a) ∧ elim 3
>
> 5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3
>
> > 6) b
> >
> > Person(b) ∧ Loves(a,b) Assume
> >
> > 7) Person(b) ∧ elim 6
> >
> > 8) Loves(a,b) ∧ elim 6
> >
> > > 9) c
> > >
> > > 10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1
> > >
> > > 11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10
> > >
> > > 12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11
> > >
> > > > 12) Person(c) Assume
> > > >
> > > > > 13) e
> > > > >
> > > > > > 14) Person(e) Assume
> > > > > >
> > > > > > 15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) ∀ elim 1
> > > > > >
> > > > > > 16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) → elim 12,15
> > > > > >
> > > > > > 17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e))

∀ elim 16

18) Person(e) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y))

∀ elim 1

19) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y)

→ elim 14, 18

20) (∃z(Person(z) ∧ Loves(a, z)) → Loves(e,a) ∀ Elim 19

.

.

fw) ∃z(Person(z) ∧ Loves(e, z)) ?

me) Loves(c,e) → elim 17, fw

sl) Person(e) → Loves (c,e) → Intro 14-me

im)∀y(Person(y) → Loves (c,y)) ∀ Intro, 13-sl

dp) (Person(c) → ∀y(Person(y) → Loves (c,y))) ∀ Intro, 12-im

bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∀ intro 9-dp

jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 5, 6-bs

Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 2, 3-jp

NOW, we just need to know the antecedent of 20, that a loves some person, or

∃z(Person(z) ∧ Loves(a, z))

But we already went through that. a loves b (lines 7 and 8). So we can add them together and existentially quantify over b:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) — Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) — Premise

3) a (Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) — Assume

4) Person(a) — ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) — ∧ elim 3

6) b Person(b) ∧ Loves(a,b) — Assume

7) Person(b) — ∧ elim 6

8) Loves(a,b) — ∧ elim 6

9) c

10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) — ∀ Elim 1

11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) — →elim 7, 10

12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) — ∀ Elim 11

12) Person(c) — Assume

13) e

14) Person(e) — Assume

15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) — ∀ elim 1

16) $\forall y(\exists z(Person(z) \land Loves(y, z)) \rightarrow Loves(c,e))$

$\rightarrow$ elim 12,15

17) $\exists z(Person(z) \land Loves(e, z)) \rightarrow Loves(c,e))$

$\forall$ elim 16

18) $Person(e) \rightarrow \forall y(\exists z(Person(z) \land Loves(y, z)) \rightarrow Loves(e,y))$

$\forall$ elim 1

19) $\forall y(\exists z(Person(z) \land Loves(y, z)) \rightarrow Loves(e,y)$

$\rightarrow$ elim 14, 18

20) $(\exists z(Person(z) \land Loves(a, z)) \rightarrow Loves(e,a)$ — $\forall$ Elim 19

21) $Person(b) \land Loves(a, b)$ — $\land$ Intro 7, 8

22) $\exists z(Person(z) \land Loves (a,z)$ — $\exists$ Intro 21

.

.

fw) $\exists z(Person(z) \land Loves(e, z))$ — ?

me) $Loves(c,e)$ — $\rightarrow$ elim 17, fw

sl) $Person(e) \rightarrow Loves (c,e)$ — $\rightarrow$ Intro 14-me

im) $\forall y(Person(y) \rightarrow Loves (c,y))$ — $\forall$ Intro, 13-sl

dp) $(Person(c) \rightarrow \forall y(Person(y) \rightarrow Loves (c,y)))$ — $\forall$ Intro, 12-im

bs) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ — $\forall$ intro 9-dp

jp) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ — $\exists$ elim, 5, 6-bs

Conc) $\therefore \forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ — $\exists$ elim, 2, 3-jp

A quick $\rightarrow$ elim of 20 and 22:

1) ∀x(Person(x) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(x,y)) Premise

2) ∃x(Person(x) ∧ ∃y(Person(y) ∧ Loves(x, y))) Premise

3)

a

(Person(a) ∧ ∃y(Person(y) ∧ Loves(a, y)) Assume

4) Person(a) ∧ elim 3

5) ∃y(Person(y) ∧ Loves(a, y)) ∧ elim 3

6)

b

Person(b) ∧ Loves(a,b) Assume

7) Person(b) ∧ elim 6

8) Loves(a,b) ∧ elim 6

9)

c

10) Person(b) → ∀y(Ez(Person(z) ∧ Loves(y, z)) → Loves(b,y) ∀ Elim 1

11) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(b,y) →elim 7, 10

12) (∃z(Person(z) ∧ Loves(c, z)) → Loves(b,c) ∀ Elim 11

12) Person(c) Assume

13)

e

14) Person(e) Assume

15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) ∀ elim 1

16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) → elim 12,15

17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e)) ∀ elim 16

18) Person(e) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y)) ∀ elim 1

19) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y) → elim 14, 18

20) (∃z(Person(z) ∧ Loves(a, z)) → Loves(e,a) ∀ Elim 19

21) Person(b) ∧ Loves(a, b) ∧ Intro 7, 8

22) ∃z(Person(z) ∧ Loves (a,z) ∃ Intro 21

23) Loves(e,a) → elim 20, 22

.

.

fw) ∃z(Person(z) ∧ Loves(e, z)) ?

me) Loves(c,e) → elim 17, fw

sl) Person(e) → Loves (c,e) → Intro 14-me

im)∀y(Person(y) → Loves (c,y)) ∀ Intro, 13-sl

dp) (Person(c) → ∀y(Person(y) → Loves (c,y))) ∀ Intro, 12-im

bs) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∀ intro 9-dp

jp) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 5, 6-bs

Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 2, 3-jp

Then we are home free. Remember that in (fw) we want

∃z(Person(z) ∧ Loves(e, z))

now, a is that person z, so do a quick ∧ Intro, and then (fw) is an ∃ Intro, and we are done.

1) $\forall x(Person(x) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(x,y))$	Premise
2) $\exists x(Person(x) \wedge \exists y(Person(y) \wedge Loves(x, y)))$	Premise
3) a (Person(a) $\wedge$ $\exists y(Person(y) \wedge Loves(a, y))$	Assume
4) Person(a)	$\wedge$ elim 3
5) $\exists y(Person(y) \wedge Loves(a, y))$	$\wedge$ elim 3
6) b Person(b) $\wedge$ Loves(a,b)	Assume
7) Person(b)	$\wedge$ elim 6
8) Loves(a,b)	$\wedge$ elim 6
9) c	
10) Person(b) $\rightarrow$ $\forall y(Ez(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$	$\forall$ Elim 1
11) $\forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$	$\rightarrow$elim 7, 10
12) $(\exists z(Person(z) \wedge Loves(c, z)) \rightarrow Loves(b,c)$	$\forall$ Elim 11
12) Person(c)	Assume
13)	

e

14) $Person(e)$ Assume

15) $Person(c) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(c,e))$ $\forall$ elim 1

16) $\forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(c,e))$ $\rightarrow$ elim 12,15

17) $\exists z(Person(z) \wedge Loves(e, z)) \rightarrow Loves(c,e))$ $\forall$ elim 16

18) $Person(e) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(e,y))$ $\forall$ elim 1

19) $\forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(e,y)$ $\rightarrow$ elim 14, 18

20) $(\exists z(Person(z) \wedge Loves(a, z)) \rightarrow Loves(e,a)$ $\forall$ Elim 19

21) $Person(b) \wedge Loves(a, b)$ $\wedge$ Intro 7, 8

22) $\exists z(Person(z) \wedge Loves (a,z)$ $\exists$ Intro 21

23) $Loves(e,a)$ $\rightarrow$ elim 20, 22

24) $Person(a) \wedge Loves(e,a)$ $\wedge$ Intro 4, 23

25) $\exists z(Person(z) \wedge Loves(e, z))$ $\exists$ Intro 24

me) $Loves(c,e)$ $\rightarrow$ elim 17, fw

sl) $Person(e) \rightarrow Loves (c,e)$ $\rightarrow$ Intro 14-me

im) $\forall y(Person(y) \rightarrow Loves (c,y))$ $\forall$ Intro, 13-sl

dp) $(Person(c) \rightarrow \forall y(Person(y) \rightarrow Loves (c,y)))$ $\forall$ Intro, 12-im

bs) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ $\forall$ intro 9-dp

jp) $\forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ $\exists$ elim, 5, 6-bs

Conc) $\therefore \forall x(Person(x) \rightarrow \forall y(Person(y) \rightarrow Loves (x,y)))$ $\exists$ elim, 2, 3-jp

And, done! Now just fill in the metal bands with Roman numerals:

1) $\forall x(Person(x) \rightarrow \forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(x,y))$	Premise
2) $\exists x(Person(x) \wedge \exists y(Person(y) \wedge Loves(x, y)))$	Premise
3) a	
$(Person(a) \wedge \exists y(Person(y) \wedge Loves(a, y))$	Assume
4) $Person(a)$	$\wedge$ elim 3
5) $\exists y(Person(y) \wedge Loves(a, y))$	$\wedge$ elim 3
6) b	
$Person(b) \wedge Loves(a,b)$	Assume
7) $Person(b)$	$\wedge$ elim 6
8) $Loves(a,b)$	$\wedge$ elim 6
9) c	
10) $Person(b) \rightarrow \forall y(Ez(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$	$\forall$ Elim 1
11) $\forall y(\exists z(Person(z) \wedge Loves(y, z)) \rightarrow Loves(b,y)$	$\rightarrow$elim 7, 10
12) $(\exists z(Person(z) \wedge Loves(c, z)) \rightarrow Loves(b,c)$	$\forall$ Elim 11
12) $Person(c)$	Assume

13) e

14) Person(e) Assume

15) Person(c) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) ∀ elim 1

16) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(c,e)) → elim 12,15

17) ∃z(Person(z) ∧ Loves(e, z)) → Loves(c,e)) ∀ elim 16

18) Person(e) → ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y)) ∀ elim 1

19) ∀y(∃z(Person(z) ∧ Loves(y, z)) → Loves(e,y) → elim 14, 18

20) (∃z(Person(z) ∧ Loves(a, z)) → Loves(e,a) ∀ Elim 19

21) Person(b) ∧ Loves(a, b) ∧ Intro 7, 8

22) ∃z(Person(z) ∧ Loves (a,z) ∃ Intro 21

23) Loves(e,a) → elim 20, 22

24) Person(a) ∧ Loves(e,a) ∧ Intro 4, 23

25) ∃z(Person(z) ∧ Loves(e, z)) ∃ Intro 24

26) Loves(c,e) → elim 17, 25

27) Person(e) → Loves (c,e) → Intro 14-26

28) ∀y(Person(y) → Loves (c,y)) ∀ Intro, 13-27

29) (Person(c) → ∀y(Person(y) → Loves (c,y))) ∀ Intro, 12-28

30) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∀ intro 9-29

31) ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 5, 6-30

Conc) ∴ ∀x(Person(x) → ∀y(Person(y) → Loves (x,y))) ∃ elim, 2, 3-31

Post-Mortem on this really hard proof:

Everything went as it was more or less forced to up until line 13. We set up subproofs for ∃ elims, ∀ intros, and → intros. After 13, we worked backwards from what became line 25. We used what we knew we wanted in line 25 to start instantiating our universals, and kept that reasoning up. The doubly universally quantified consequent in line 1 was the key, and we took advantage of the fact that basically anything could ultimately be an instance of that. Then all that remained was to keep using our subgoals to help us decide what names to instantiate for what universally quantified clauses and sentences.

That's all. Sort of hides all the pain and mental anguish of actually doing it, doesn't it? Anyway, that is one of the hardest proofs I know. They could get more complicated by having 3-place relations and triple chains of relations (in what we have, there is ultimately a double chain, where one lover is loved by another lover), but the principles would remain the same even if the proof got longer.

Chapter What

Whither Logic?

Well. That is just about it for basic formal logic. There's some stuff we didn't really go over – functors (relation like functions that take singular terms as arguments, and give terms, not truth values, as outputs), the logic of identity (pretty straightforward, actually – captured by a few rules that govern the relation of identity), some translation issues – but by and large, we've covered all of predicate logic.

So is that it? Are we at another point like Kant thought, where logic is complete? Far from it! I rather think that one of the tasks of logic is to systematically model natural languages (the more you model, the more arguments you can formalize), and there is just a *ton* of philosophical work to be done even in ordinary Boolean logic! For example, there is still a wide-open debate about how to treat the material conditional if-then. Many if not most philosophers think that the usual, truth-functional treatment of if-then is inadequate, and that it cannot really faithfully model the implication relation that we, as philosophers, are interested in when we make arguments. There are many competing theories on how to resolve the so-called "paradoxes of the material conditional" that we hinted at in volume one. Even something as prosaic as 'or' has its philosophical issues and corresponding debates. Often, a good deal of metaphysics is rolled out to try and resole these issues.

Although we have a logic of predicates and relations, which neatly maps onto adjectives, we have no similar logic for adverbs. Where adjectives / predicates modify the subject (a singular term), adjectives modify adverbs / predicates. And they don't seem to do so in any easily systematic way: A very tall person is still tall, but a decoy duck is not a duck at all (and as one former president tried to argue, oral sex is not sex). Moreover, there is a continuing debate about the natural language universal quantifier: does it imply existence? Does 'All men are mortal' imply that there are some mortal men? In standard predicate logic, it does not; but there may be some compelling arguments that in NL it does imply existence, and the standard treatment is incorrect.

And that's just for basic formal logic. There are vast fragments of natural language that we haven't formalized, some of which are of deep importance to philosophers. How do we model time – is it a predicate of events, like tenses? Should we use 'x is past' 'x is future', 'x is past in the future', and so on? Or is it better expressed as a relation, with no primitive tenses – 'x is earlier than y', 'y is contemporaneous with x', and so on? What rules go with these? Each strategy has different philosophical (and even scientific) commitments, and within each strategy, we can model different claims about time. Is it one-way only? Does it, as Nietszche claims (albeit metaphorically), loop back and eternally recur?

There is a good, brief, accessible discussion of decidability issues (on which I have drawn) in Graeme Forbes, *Modern Logic* (Oxford, 1994), pp. 207-211, 283-286. The result that in monadic predicate logic (without identity), an invalid argument will have a counterexample of size 2*k* is proven in Boolos, Burgess, and Jeffrey, *Computability and Logic*, Fourth Edition (Cambridge University Press, 2002), section 21.2.