Formal Languages and Logic

Daniel W. Harris

February 23, 2017

A formal (or artificial) language is a language invented by humans, and whose properties are precisely stipulated by us. Formal languages stand in contrast to natural languages, such as English and Swahili, which, although they arise from human activity, are not invented from scratch by any particular human or group of humans, but have rather evolved over time as a result of a mixture of human biology and culture. One big difference between natural and formal languages is what they are used for. Whereas natural languages are used by humans to communicate with one another, either by speech, writing, signing, or in some other medium, formal languages are often used for mathematical or scientific purposes, or to interact with computers.[1]

Some formal languages that you may have encountered already include the notation of mathematics. Here are some sentences of the formal languages that you may have learned to use in math courses, for example:

(1) $(x + 7) \times 9 = y$

(2) $\int \frac{d\theta}{1+\theta^2} = \tan^{-1}\theta + C$

And here are some examples of formal languages (specifically, Python and C++) that you may have learned to use in computer science courses:

(3)
```
print 'Hello, world!'
```

(4)
```
int main()
   {
   std::cout << "Hello World!";
     std::cout << std::endl;
     return 0;
     }
```

[1] Constructed languages like Esperanto are borderline cases. Esperanto was created by a human (L. L. Zamenhof), but it resembles natural languages in various respects and is used by humans to communicate. In any case, there are good reasons to think that most natural languages did not arise in this way.

The formal languages used most often by philosophers are the languages of propositional and first-order logic, sometimes with some complications added to them. This is an introductory guide to formal languages and logic designed for students in my course on Bertrand Russell in Spring 2017. This guide is brief and not particularly rigorous, and although it covers some of the same material that would be covered in a formal logic course, it is not a replacement for such a course. The purpose of it is to help you to quickly become familiar with the features of formal languages as they're used by philosophers. Specifically, the goals of this guide are as follows:

- To introduce you to the idea of formal languages and formal logic (for those who haven't encountered them before).
- To help you learn to understand sentences of a couple of formal languages that often show up in philosophy essays and books.
- To make it easier to understand some of the ways that Bertrand Russell used formal languages when doing philosophy.

1 A Word About Notational Conventions

One thing that makes formal logic difficult to read is that different texts use different notation and terminology for the same things. The notations used by Frege and Russell looked entirely different from each other, and both notations are different than contemporary standards (though contemporary notation is much more similar to Russell's than to Frege's). Consider the sentence that we would normally read out as 'every F is a G', for example. (Here, 'F' and 'G' stand in arbitrarily for predicate expressions, such as 'human' and 'mammal' or 'natural number' and 'rational number'.) Contemporary textbooks would annotate this sentence in one of the following ways.

(5) $(\forall x)(Fx \supset Gx)$

(6) $(\forall x)(Fx \rightarrow Gx)$

(7) $(x)(Fx \supset Gx)$

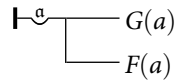
(8) $(x)(Fx \rightarrow Gx)$

Russell would have written the same sentence in one of the following ways (the second is his way of abbreviating the first).

(9) $(x)Fx \supset Gx$

(10) $Fx \supset_x Gx$

And Frege, whose annotation included two dimensional, graphical elements, would have written this sentence as follows:

(11) 

And of course, there are other options. If you read work by any of the great Polish logicians of the mid-Twentieth Century, for example, you might see the above sentence represented in Polish notation, like this:

(12) $PxCFxGx$

And if you read work by contemporary linguists or philosophers who work on natural-language semantics, you might see the sentence written as follows.

(13) [Every F : Fx]Gx

There are complicated historical reasons for these different notational conventions. In what follows, I'll attempt to give a sense of some of the different options, while mostly focusing on the contemporary versions that you're likely to encounter when reading philosophy these days. A good way to become really comfortable with the mess of terminologies that logicians use is to look through several different logic textbooks and compare their different conventions.

2 Why Formal Languages?

A lot of philosophy students wonder about things like this: *Why do philosophers need all these symbols anyway? Don't philosophers use logic mainly to make themselves feel like they're doing something scientific when really they're just playing with letters? What's the point?* Well, it's certainly true that philosophers sometimes use formalism in unnecessary and perhaps even onanistic ways. But that's certainly not always the case, and once you learn the symbols you may find that they can make certain arguments and claims much easier to understand. Once you become familiar with a couple of formal languages, you'll see that they are much simpler than natural languages in several important respects. Most importantly, the sentences of formal languages have simple sentence structures that follow simple and precise stipulations.

These qualities of formal languages have proven useful to philosophers for several reasons. The structures and meanings of formal-language sentences are well understood, and they are designed to eliminate some features of natural language, such as ambiguity and vagueness, that can get in the way of precise understanding. Philosophers therefore sometimes translate their claims and arguments into formal language in order to make them more precise. As we'll see below, this clarity and precision is especially useful in logic, whose goal is to state the rules by which some sentences follow logically from other sentences. (For example, the sentence 'Bert is mortal' follows from the sentences 'Every human is mortal' and the sentence 'Bert is human'.)

Formal languages have also been used by philosophers and linguists to understand the principles that guide natural languages. There are good reasons to think that humans have a precise and systematic way of understanding the meanings of sentences on the basis of the meanings of words. But because natural languages are so complex, it's very difficult to figure out this ability of works. So, semanticists (who may be philosophers, linguists, psychologists, or computer scientists) create and study formal languages instead, using them as simplified models, and building in more of the features of natural languages as they go. You can think of this as the process of reverse-engineering an algorithm that runs unconsciously in the human mind.

Beginning with early analytic philosophers like Gottlob Frege, Bertrand Russell, and Ludwig Wittgenstein, many philosophers have also taken the study of formal languages to be crucial for solving longstanding philosophical problems in metaphysics and epistemology. Consider, for example, Russell's book, *Our Knowledge of the External World*, whose second chapter is called 'Logic as the Essence of Philosophy'. One reason why Frege and Russell gave logic such a central place in their work is that they thought that all of mathematics could be deduced from a few logical axioms together with a small number of logical inference rules. If this were true, it would be a big deal, because it would make progress on some very tough philosophical puzzles about how we know mathematical truths and what the subject matter of mathematics is—problems that has beguiled philosophers since Plato. Russell and Alfred North Whitehead attempted to work through the minute details of reducing math to logic in their 2000-page opus, *Principia Mathematica*. In the introduction to that book (pp.1–3), Russell & Whitehead list off a whole range of excellent reasons why it made more sense to carry out their task using the symbolism of their artificial language. And since he took it to be so useful in the philosophy of mathematics, Russell went looking for other applications as well.

Eventually Russell came to think that the most important task for philosophers was to learn how to properly translate our knowledge about the world into a precise formal language, thus dissolving a variety of puzzles and revealing the hidden structure of reality (as mirrored by the structure of his formal language itself).

Many later analytic philosophers have agreed with Russell about the centrality of logic to philosophy, and many of the most influential philosophers of the 20th Century have also done some of their work using formal languages. Even if they're wrong about the importance of logic to philosophy, you'll still need to be able to read the formal languages they use in order to make sense of their writings.

2.1 Logic

Philosophers often study formal languages in order to study logic, and we'll follow this format here. Let me say a bit more about what logic is, and why it needs to be formal.

The aim of logic is to say which sentences of a language *follow from* which other sentences of a language. If the object language[2] is English, for example, then we can say that the sentence 'someone knows logic' follows from the sentence 'Bert knows logic'. There are lots of different ways in which we could state this fact, including these:

- 'Someone knows logic' is a (logical) consequence of 'Bert knows logic'
- 'Bert knows logic' entails 'Someone knows logic'
- 'Someone knows logic' can be validly inferred from 'Bert knows logic'

We can also say that our example is an instance of the *consequence relation* in English. A consequence relation is a relation that holds between a pair of sentences just in case the second follows from the first. A logic is a theory that characterizes the consequence relation for a given language.

There are lots of different ways of characterizing a consequence relation. They can be grouped into two categories: *proof-theory* and *semantics*.

2.2 Proof Theory

Broadly speaking, proof-theories are made up of formal inference rules—rules that ignore what words and sentences mean and take only their forms into account in saying what follows from what. A proof-theoretic characterization of consequence will say that one sentence follows from another if and only if the former can be derived from the latter solely through the application of a series of these formal inference rules. If we wanted an inference rule that could be use to derive entailments like the example about Bert knowing logic, for example, we might try this one:

(14) EXISTENTIAL GENERALIZATION (ENGLISH)
Whenever you have an English sentence with the name of of a person or a thing, you may replace the name with either 'someone' or 'something', respectively.

This isn't a very good formal inference rule for several reasons. One problem is that in order to use this rule reliably, we need a precise definition of what a name is. It seems clear enough that 'Bert' is a name, but what if my sentence is 'Bert Russell knows logic'? Is 'Bert' a name in this sentence, or only part of a name? What about when the sentence is 'Bertrand knows logic'? In this case it is even more tempting to say that 'Bert' is only part of a name, but there is nothing in our inference rule that says how to make this distinction. And what about names like 'John Doe', and definite descriptions like 'the president of the United States'? What about names for fictional characters like 'Bart Simpson'? Should these all count as names for the purposes of our inference rule? All of these troublesome cases show that inference rules need to rest on very careful definitions if we want them to be universal and precise. Definitions of this kind are very difficult to give when we're talking about natural languages like English, and this is one reason why logicians tend to focus on artificial languages with simple and precise sentence structures.

Another problem with our inference rule is that it is not entirely formal, since, in order to apply it, we need to know whether the name in question stands for a person or a thing. For example, suppose I have a boat that I named 'Mary', after my mother. Now suppose that we want to apply our inference rule to the sentence 'Mary is at the yacht club'. In order to know whether to substitute 'something' or 'someone' for 'Mary' in this sentence, I would need to know whether we're talking about the person Mary (my mother) or the thing Mary (my boat). If I don't get this part right, I might draw the conclusion that someone is at the yacht club from a premise about my boat, which would be a bad inference. But this means that I would have to know something about the name's meaning, and so that inference rule isn't truly formal. This illustrates just one of the many reasons why it is very difficult to give fully formal inference rules for English sentences, and this is another reason that logicians tend to focus on formal languages.

To sum up: if we want to give a proof theory for a language, we first need a precise understanding of the language's syntax—the rules by which its primitive symbols (roughly, its words) combine into sentences. Although linguists have come a long way toward gaining this understanding when it comes to natural languages like English, they aren't all the way there yet, and the results are extremely complicated. But I'll give examples of a couple of proof-theories for formal languages below.

[2] **Object Language vs. Metalanguage:** When studying languages—particularly the artificial languages in which logics are framed—it is often useful to distinguish the object language, which is the language being studied, from the metalanguage, which is the language in which the studying is being done. Sometimes these happen to be the same, as on the left, where I'm making claims about English using English. But other times we might use English as a metalanguage to study some other object language.

2.3 Semantics

Semantic theories characterize the consequence relation using concepts like meaning, reference, and truth. Here are three slightly different semantic definitions of consequence, for example:

(i) A sentence B follows from a set of sentences {X,Y,Z…} if and only if the truth of all of X,Y,Z…would guarantee the truth of B.

(ii) A sentence B follows from a set of sentences {X,Y,Z…} if and only if B is true in every possible situation in which all of X,Y,Z…are true.

(iii) A sentence B follows from a set of sentences {X,Y,Z…} if and only if, no matter what the non-logical vocabulary in X,Y,Z…and B mean, if all of X,Y,Z…are true then B is true.

One or more of these may look familiar if you've taken an introductory logic course or if you've studied the concept of a valid argument in another philosophy course. A valid argument is just an instance of the consequence relation, and we typically explain the concept in a semantic way, like (i). But these characterizations still aren't very precise. To really understand (i), we would need to know what it means for one sentence to guarantee the truth of another. To really understand (ii), we would need to know what a possible situation is. And to really understand (iii), we would need to know how to distinguish between logical and non-logical vocabulary. The definitions of consequence we study below will precisify these notions in different ways.

Moreover, all three definitions presuppose that we can give a precise characterization of how the truth conditions of two sentences may be systematically connected. The idea of *truth conditions* is important in both logic and the philosophy of language, where it is often assumed that something important about the meaning of a sentence is captured by its truth conditions. On an intuitive level: if you know what the world would have to be like for the sentence to be true, you know what the sentence means (or at least an important component of what it means). So, a semantic definition of the consequence relation for a whole language would have to include a theory that tells us the truth condition of every sentence in the language. But this raises a problem, because both natural and formal languages contain infinitely many sentences. This is because such languages are recursive. Consider the following sentence.

(15) My mother's sister's brother's friend's cousin's dog's chew toy is delicious.

Although it's a bit difficult to read this sentence, it's obviously a grammatical and meaningful sentence. And it seems clear that we could add in as many possessives as we want, and we'd get an even more complicated, but still grammatical and meaningful sentence with different truth conditions. The same is true of sentences in formal languages. The following sentence of the language of propositional logic (what I'll call 'PL' below) can be extended arbitrarily many times by adding '$X \rightarrow$ (' to the left side and ')' to the right side, for example.

(16) $P \rightarrow (Q \rightarrow (R \rightarrow (S \rightarrow T)))$

Each of these sentences would have different truth conditions. This means that a definition of logical consequence for either English or PL would have to be recursive: it would have to define the truth conditions of whole sentences systematically in terms of their parts, some of which might themselves be sentences. Although natural-language semanticists are working on this when it comes to languages like English, they haven't finished, and it is a very difficult task! Things are much simpler for formal languages, as we'll see below.

3 Syllogistic

Until the 19th Century, the most influential logics were based on Aristotle's syllogistic. This is the kind of logic that one studied at medieval universities, for example, and it's the kind of logic that Kant worked with.

The syllogistic works with a restricted artificial language that contains, in its purest form, only four kinds of sentence:

(17) UNIVERSAL AFFIRMATION
Every A is B.

(18) UNIVERSAL DENIAL
No A is B.

(19) PARTICULAR AFFIRMATION
Some A is B.

(20) PARTICULAR DENIAL
Some A is not B.

In each of these sentences 'A' and 'B' are variables that stand in for expressions that stand for categories, such as 'dog' or 'person who is blind'. In each sentence 'A' is the subject and 'B' is the predicate. Some extended syllogistic logics include other sentence forms, including some with singular subjects:

(21) x is B

Here, 'x' stands in for a name, such as 'Socrates'. Still, the assumption is that every sentence dealt with by syllogistic logic has a subject and a predicate.

A syllogism is an argument with two premises and a conclusion, each of which takes one of the above forms. For example:

(22) Every dogs is a mammal.
Every mammal breathes.
Every dog breathes.

This syllogism takes the following form (nicknamed 'Barbara' by medieval logicians):

(23) BARBARA
Every A is B.
Every B is C.
Therefore, every A is C.

It's relatively clear that any argument that takes this form—i.e., any argument we can get by replacing 'A', 'B', and 'C' with category expressions in a uniform way—will be valid. This makes Barbara is a formally valid inference form. Aristotle and the logicians who worked in his tradition developed systematic syllogistic logics that distinguish the valid syllogism forms from the invalid ones.

There is much more to be said about syllogistic logic. (It was, after all, the main form of logic that people studied for about 2000 years.) But I won't dwell on it here since its main interest for present purposes is as a background to contemporary logic.

Why don't we still use the syllogistic logic? The main reason is that many valid inferences can't be represented as syllogisms. Take this one, for example:

(24) If all dogs are mammals, then Fido is a mammal.
All dogs are mammals.
Therefore, Fido is a mammal.

Intuitively, this and any other argument of the same form is valid. However, the first premise of this argument doesn't have a subject–predicate structure. Instead, it has a conditional structure, whose parts are entire sentences. Since syllogistic logic assumes that premises and conclusions of arguments are in subject–predicate form, it can't handle arguments like this one. To do that, we need a logic with a different kind of language whose sentences can have sentences as parts. That's what we'll look at next.

4 Propositional Logic

Propositional logic (a.k.a. sentential logic) characterizes the consequence relation for a very simple formal language which we can call PL (for 'Propositional Language'; some textbooks call it SL for 'Sentential Language'). The basic idea of PL is that it treats simple sentences as indivisible units, ignoring all their internal grammatical structure. For example, we might translate 'Bert knows logic' as the symbol 'P'.

4.1 The Syntax of PL

To define PL, the first thing we do is to give a list of all of the symbols that make up the language:[3,4]

capital letters for atomic sentences P, Q, R, P_1, P_2, Q_1, Q_2,…
the negation symbol $\neg$
the conjunction symbol $\wedge$
the disjunction symbol $\vee$
the conditional symbol $\supset$
the biconditional symbol $\equiv$
parentheses $(,)$

We next give the syntax (a.k.a. grammar or well-formedness rules) for PL by giving a precise definition of which strings of the symbols we've enumerated count as sentences (a.k.a. well-formed formula, or *wff*).

(I) Every sentence letter P, Q, R, P_1, P_2, Q_1, Q_2,…is a sentence of PL.

(II) For any two sentences of PL, φ and ψ, the following are also sentences of PL:

(i) $\neg\varphi$
(ii) $(\varphi \wedge \psi)$
(iii) $(\varphi \vee \psi)$
(iv) $(\varphi \supset \psi)$

[3] It is overwhelmingly common to use P, Q, and R (or *p*, *q*, and *r*) to stand in for sentences when only a few letters are needed, and you will likely have encountered this practice before—a really dorky example is David Chalmers' compilation of proofs that *p*. Other capital letters and lowercase letters are also sometimes used, as are Greek letters (α, β, φ, ψ . . .) In defining a formal language, it is important to ensure that there be a potentially infinite number of sentences, and so subscripted numbers are typically added for this purpose.

[4] It is common to use $\sim$ instead of $\neg$ for negation, & or $\bullet$ instead of $\wedge$ for conjunction, $\rightarrow$ instead of $\supset$ for the conditional, and $\leftrightarrow$ instead of $\equiv$ for biconditional. There are even some alternatives to the parentheses. For example, Russell never used parentheses, but instead had a significantly more confusing system of dots that served the same purpose. Instead of writing $((P \supset Q) \supset (Q \supset P))$, for example, Russell and Whitehead would have written $P \supset Q .\supset. Q \supset P$. Here, the dots flanking the middle $\supset$ indicate that it has widest scope. If you want to know more about how Russell's notation differs from contemporary notation, have a look at Bernard Linksy's entry in the Stanford Encyclopedia of Philosophy on 'The Notation of *Principia Mathematica*'.

(v) $(\varphi \equiv \psi)$

(III) No other string of symbols is a sentence of PL.

Clause (I) defines the atomic sentences, which are the simplest sentences of PL. All of the more complex sentences of PL are constructed from atomic sentences by means of the sub-clauses of (II). Clause (II) is the most complex and important of the three clauses, and it is important to notice a couple of things about it. First, notice that the two Greek letters φ and ψ are variables that range over any sentences. They might stand for two different atomic sentences, such as P and Q (or P and P—they needn't stand for distinct sentences). If they stand for P and Q, then sub-clause (II.ii) tells us that $(P \wedge Q)$ is also a sentence. But since that's a sentence, φ and ψ range over *it*, too. So, taking φ to stand for $(P \wedge Q)$ and ψ to stand for Q, sub-clause (II.ii) tells us that $((P \wedge Q) \wedge Q)$ is also a sentence. Clauses (IIi–v) are *recursive*, which is roughly to say that their outputs can be fed back into them as inputs. It follows that with just the rules we've outlined above, even a single atomic sentence (say, P) can be repeatedly combined with itself to create an infinite number of complex sentences of PL.

Every complex sentence of PL has a main connective, which is said to *take scope* over all of the other connectives. If we imagine that the sentence was constructed by a series of applications of the rules (II.i–v), the main connective would be the most recent one to have been added. Each type of sentence is named after its main connective. For example, $\neg(P \vee Q)$ is a negation. It's not a disjunction because the negation symbol $\neg$ has wider scope than the disjunction symbol $\vee$. (But we could call it a negation of a disjunction, or a negated disjunction.)

One thing you should be able to do is look at a string of symbols and decide whether it is a sentence of PL or not, and what kind of sentence it and its parts are. This might take some practice.

4.2 Interpreting Sentences of PL

Now that we've got our simple language PL defined, let's talk a little about how to interpret it. A very important distinction to make is between the logical and non-logical vocabulary. The logical vocabulary of PL are the sentence connectives: $\neg$, $\wedge$, $\vee$, $\supset$, and $\equiv$. They can be intuitively translated as 'not', 'and', 'or', 'if…then', and 'if and only if', respectively.

Our logical vocabulary items are called connectives because you can connect them to one or two sentences to create a new sentence, as in subclauses (II.i–v) of our syntactic rules from the last section. $\neg$ is a *unary* connective because it combines with one sentence at a time, and $\wedge$, $\vee$, $\supset$, and $\equiv$ are *binary* connectives because they combine with two sentences at a time. The two subsentences in a conjunction are its *conjuncts*, the two subsentences in a disjunction are its *disjuncts*, and the first and second subsentences in a conditional are (respectively) its *antecedent* and *consequent*.

	Symbol	Intuitive English translation	Equivalent Symbols
Negation	$\neg P$	It is not the case that P	$\sim P$
Conjunction	$(P \wedge Q)$	P and Q	$(P \,\&\, Q)$, $(P \bullet Q)$
Disjunction	$(P \vee Q)$	P or Q	
Conditional	$(P \supset Q)$	If P then Q, P only if Q	$(P \rightarrow Q)$
Biconditional	$(P \equiv Q)$	P if and only if Q	$(P \leftrightarrow Q)$

Table 1: Connectives

Sentence letters like P and Q are non-logical vocabulary. Whereas each item of logical vocabulary has the same meaning every time you see it, the sentence letters are variables, which means that they stand in for arbitrary sentences with arbitrary meanings. If we want to translate some English into PL, we can stipulate which English sentences will be symbolized with which letters, but we can later reuse the same letters to symbolize other sentences of English. Table 2 contains some examples of translations from English into PL, using each sentence letter to translate the same sentence of English throughout.

The ability to translate back and forth from English into formal languages like PL is one of the most useful logic skills to have as a philosopher, because many philosophers throw bits of standard logical notation into their work without explaining how to read it. But if you can stare at a sentence of PL for a few seconds and translate it to yourself, you'll have a much easier time with those passages. (This is even more true of First-Order notation, which is even more common, and which we'll look at after PL.)

Of course, translating into a even a simple language like PL can get much more complicated than these examples suggest, and it is a skill that is best acquired through practicing with lots of examples. It would be a good idea to read through a chapter of a logic textbook that explains some techniques, and then to work through the exercises at the end of the chapter. Here are some options:

- *A Modern Formal Logic Primer* by Paul Teller, Ch.2, pp.21–28
- *An Introduction to Formal Logic* by Peter Smith, ch.10, pp.82–87

4.3 Semantics for PL

Remember the three semantic definitions of logical consequence that I mentioned earlier:

English Sentence	PL Translation
Russell invented logic.	R
Frege invented logic.	F
Aristotle invented logic.	A
Russell invented logic and Frege invented logic.	$R \wedge F$
If Frege invented logic, then Frege did not invent logic.	$F \supset \neg R$
If Aristotle invented logic, then Frege did not invent logic and Russell did not invent logic.	$A \supset (\neg F \wedge \neg R)$
Either Aristotle invented logic or Frege invented logic.	$A \vee F$
It is not the case that Aristotle invented logic, Frege invented logic, and Russell invented logic.	$\neg(A \wedge (F \wedge R))$

Table 2: Some Translations into PL

(i) A sentence B follows from a set of sentences {X,Y,Z…} if and only if the truth of all of X,Y,Z…would guarantee the truth of B.

(ii) A sentence B follows from a set of sentences {X,Y,Z…} if and only if B is true in every possible situation in which all of X,Y,Z…are true.

(iii) A sentence B follows from a set of sentences {X,Y,Z…} if and only if, no matter what the non-logical vocabulary in X,Y,Z…and B mean, if all of X,Y,Z…are true then B is true.

We're now in a position to make more sense of these definitions, and to show that they all amount to the same thing, when it comes to PL. In the last subsection, we distinguished between the logical and nonlogical vocabulary of PL, and this puts us in a position to make good on (iii). The trick to this will be to give clear and precise meanings to our logical vocabulary (i.e., the connectives).

This is possible because our logical vocabulary are all connectives which are *truth-functional*. A truth-functional connective combines with one or more sentences to create a new sentence whose truth or falsity depends only on the truth or falsity of the original sentences. For example, the negation symbol $\neg$ combines with any sentence φ and has the following effect: if φ is true, then $\neg\varphi$ is false, and vice versa.[5]

We can express express the meanings of truth-functional connectives by using truth tables. For example, Table 3 is truth table for negation.

P	$\neg$P
T	F
F	T

Table 3: Negation

The other connectives work the same way, but their truth tables need more rows and columns because they are binary connectives. They are presented in Table 7.

P	Q	P $\wedge$ Q
T	T	T
T	F	F
F	T	F
F	F	F

(a) Conjunction

P	Q	P $\vee$ Q
T	T	T
T	F	T
F	T	T
F	F	F

(b) Disjunction

P	Q	P $\supset$ Q
T	T	T
T	F	F
F	T	T
F	F	T

(c) Conditional

P	Q	P $\equiv$ Q
T	T	T
T	F	F
F	T	F
F	F	T

(d) Biconditional

Table 4: Truth Tables for Binary Connectives

Table 4a says that, for a conjunction to be true, both of its conjuncts have to be true. This seems right: for 'John smokes and Tim dances' to be true, 'John smokes' and 'Jane dances' both have to be true as well. Table 4b says that for a disjunction to be true, at least one of its disjuncts has to be true. Table 4c says that a conditional is false only when its antecedent is true and its consequent is false. Table 4d says that a biconditional is true only when both its subsentences have the same truth value.

Each of these truth tables captures the meaning of its connective—the contribution that the connective makes to the truth conditions of sentences in which it appears. It should be clear that these meanings are simpler than those of their natural-language

[5] I said earlier that the intuitive meaning of $\neg$ in English is 'not'. But of course, both the syntax and semantics of 'not' are much more complicated than those of '$\neg$'. For example, you can't just put the word 'not' at the start of an English sentence; it goes somewhere in the middle. However, nearly every sentence of English with 'not' somewhere in it can be translated into a sentence of PL with $\neg$ to the left of a complete sentence. In general, there is much debate over the relationship between logical symbols and their English counterparts. If you'd like to see what I mean, check out Lawrence Horn's book, *A Natural History of Negation*, which is a 684-page tome about negation in English. *The Geneology of Disjunction* by Raymond Jennings is a similar book about disjunction, written by my undergraduate logic professor. There have been thousands of articles and books published about conditionals. Even 'and' has been the subject of many pages of theorizing. Remember: part of the reason we do logic in artificial languages is that it makes things simpler!

counterparts in certain ways. For example, in saying 'they got married and they had a baby', part of what I would normally be communicating is that the marriage happened before the baby. But this is not captured in the meaning of ∧: P∧Q is equivalent to Q*wedge*P. Similarly, when I say 'we will have an exam on the second-last day of class or we'll have an exam on the last day of class', I typically communicate that we won't have exams on both days. But P∨Q is true when both P and Q are true—∨ is *inclusive*. In these ways, the connectives are simpler than their English counterparts.

This is particularly true when it comes to the conditional, whose truth table has some weird consequences. For example: for any two true sentences, P and Q, the truth table says that P⊃Q is also true, even if P and Q have nothing to do with one another. For example: let P stand for 'Germany is a country' and Q stand for 'The sun will eventually die'. Since these are both true sentences, our truth table says that the following sentence is also true: 'Germany is a country ⊃ the sun will eventually die.' But the corresponding sentence of English sounds false: 'If Germany is a country then the sun will eventually die'. Intuitively, the English sentence seems to imply that there is a connection between Germany's being a country and death of the sun, but there is no such connection. The same point goes for conditionals whose antecedent and consequent are both false. Consider the sentence, 'If George Bush has walked on the moon, then 9/11 was an inside job'. This sentence sounds false, or at least weird. But our truth table for ⊃ says that the following sentence is true: 'George Bush has walked on the moon ⊃ 9/11 was an inside job'. These paradoxical results are one of the reasons that so much has been written about the conditional. To make a long story short: most philosophers and linguists no longer think that the English conditional is actually a truth-functional connective at all; its meaning is more complicated in ways we needn't get into here. But again, remember: part of our reason for doing symbolic logic is that it makes things simpler! And ⊃ shares one crucial feature with the English conditional: a conditional of either kind is false if it has a true antecedent and a false consequent.

Part of what is special about logical vocabulary items, such as the connectives, is that their meanings are simple and mathematically precise enough to be captured by truth tables like these. We can think of the rows of a truth table as representing different possible situations—all of the different ways that things could be with respect to the truth or falsity of the sentence letters with which the truth table deals. In the truth tables in Table 7, for example, the top row represents a possible situation in which both P and Q are true, the second row represents a situation in which P is true and Q is false, and so on. We don't need to know what P and Q stand for in order to know that there are only four possible combinations of truth values that they could have, and because the binary connectives are truth-functional, those combinations correspond to the only four ways of dividing up all of the world's possible situations in a way that could be relevant to the truth of complex sentences of PL that contain instances of only two atomic sentences as parts.

We can also create a truth table for a more complex sentence with more than two sentence letters in it. Table 5 is an example.

P	Q	R	P ⊃ (Q ∧ R)
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	T
F	F	T	T
F	F	F	T

Table 5: Truth Tables for ⊃ (Q ∧ R)

This truth table has eight rows because there are eight possible combinations of truth values for its three sentence letters. (In general, a truth table will have 2^n rows if n is the number of distinct sentence letters involved.) But the basic principle is the same as before: each row represents a way that the world could be with respect to the truth or falsity of P, Q, and R, and the truth or falsity of P ⊃ (Q ∧ R) wholly determined by the combination of its parts' truth values.

We now have all the tools needed to semantically define a consequence relation for PL by creating a truth table for a whole argument at once. Table 6 is an example.

P	Q	Premise 1: P ⊃ Q	Premise 2: ¬Q	Conclusion: ¬P
T	T	T	F	F
T	F	F	T	F
F	T	T	F	T
F	F	T	T	T

Table 6: Truth Table for P ⊃ Q, ¬Q $\models_{PL}$ ¬P

This truth table precisifies and exemplifies each of the semantic definitions of consequence that I recalled at the beginning of this subsection. In accordance with the third definition: it shows that no matter what meanings are had by the non-logical vocabulary in the premises and conclusion (i.e., P, Q, and R), if the premises are all true, then the conclusion is true. In this truth table, the premises are all true only on the last row, and the conclusion is also true at that row. In accordance with the second definition: the conclusion is true in every possible situation (i.e., row of the truth table) at which all of the premises are true. And, in accordance with the first definition, there is a clear sense

in which the truth of all of the premises guarantees the truth of the conclusion.

More generally, we can define semantic consequence for PL like this: an argument made up of sentences of PL is valid if and only if, when we construct a truth table for it, its conclusion is true at every row of the truth table at which all of the premises are true. And we can use the $\models_{PL}$ symbol (pronounced 'double turnstile') to represent the semantic consequence relation, as I do in the caption for Table 6.

It's also possible to use a truth table to show that certain sentences of PL are true in every possible situation. These are sometimes called valid sentences—think of them as valid arguments with no premises needed. They are also sometimes called *tautologies* or *logical truths*. An example is $(P \supset (Q \vee P))$. If we were to construct a truth table for this sentence, we would find that the sentence comes out true at every row.

Constructing truth tables to test the validity of arguments and sentences is another skill that it takes practice to get good at. And if you practice enough, you'll be able to run through the process in your head without having to write anything down. This is a handy skill for a philosopher to have. To develop it, check out the following logic-textbook chapters and work through the exercises at the end:

- *A Modern Formal Logic Primer* by Paul Teller, Ch.2, pp.21–28
- *An Introduction to Formal Logic* by Peter Smith, chs.9–15, pp.72–144 (Smith mixes his semantics in with lots of other stuff.)

4.4 Proof Theory for PL

Remember: a *proof-theory* (a.k.a. a *formal system*, or a *calculus*) is a set of rules that shows how to mechanically derive the sentences of a language from other sentences of that language guided by facts about their form (i.e. syntax) only, and not paying any attention to their meanings. There are lots of different ways to give a proof theory for PL. I will quickly summarize a simple version of *natural deduction* here.

The idea behind a natural deduction system is to construct a set of rules that are relatively intuitive and easy for humans to memorize and apply. I'll explain one such set of rules in a moment; first, have a look at an example of a natural-deduction proof:

Example 1: $P \supset Q, P \vdash_{PL} Q$

1	(1)	$P \supset Q$	Premise
2	(2)	P	Premise
1, 2	(3)	Q	1,2 $\supset$E

At the top of a proof is a label with the argument, or *sequent*, that we're trying to prove. A sequent consists of a list of premises followed by the $\vdash_{PL}$ symbol, which represents the relation of provability, or proof-theoretic consequence (for PL), followed by a conclusion. Each line of the proof is made up of a numbered sentence with a list of its assumptions to its left and a justification to its right. The word 'premise' in a line's justification column signals that that the line is a premise of the argument. In Example 1, lines (1) and (2) are premises. Since they are themselves assumptions, their own line numbers are listed in the assumptions column. When a line is derived from one or more other earlier lines by means of some inference rule, this is recorded in the justification column. In Example 1, line (3) depends on lines (1) and (2) and the inference rule $\supset$E (on which more in a moment). The numbers 1 and 2 are listed in line (3)'s assumptions column because it relies on the premises at lines (1) and (2).

The trick with natural-deduction proofs is to memorize a full collection of inference rules, each of which is an instruction for what to write on the next line of a proof based on what you've already got on previous lines. A full collection typically includes an introduction rule and an elimination rule for each of the connectives, $\neg$, $\wedge$, $\vee$, $\supset$, and $\equiv$. Of the ten rules, we can write eight of them quite simply, as sequents:

$\neg$E $\neg\neg P \vdash_{PL} P$
a.k.a. negation-elimination, double negation

$\wedge$I $P, Q \vdash_{PL} P \wedge Q$
a.k.a. conjunction-introduction, and-introduction, or sometimes just conjunction

$\wedge$E $P \wedge Q \vdash_{PL} P$ (or: $P \wedge Q \vdash_{PL} Q$)
a.k.a. conjunction-elimination, and-elimination, simplification

$\vee$I $P \vdash_{PL} P \vee Q$
a.k.a. disjunction-introduction, or-introduction, addition

$\vee$E $P \vee Q, \neg P \vdash_{PL} Q$
a.k.a. or-elimination, disjunctive syllogism, modus tollendo ponens

$\supset$E $P \supset Q, P \vdash_{PL} Q$
a.k.a. conditional-elimination, implication-elimination, modus ponens, modus ponendo ponens, detachment

$\equiv$I $P \supset Q, Q \supset P \vdash_{PL} P \equiv Q$
a.k.a. biconditional-introduction

$\equiv$E $P \equiv Q \vdash_{PL} P \supset Q$ (or: $P \equiv Q \vdash_{PL} Q \supset P$)
a.k.a. biconditional-elimination

Each of these rules says that if you have lines in your proof that correspond to the form of the premises, then you can write a new line corresponding to the form of the conclusion. In doing so, you must cite the lines corresponding to the premises together with the inference rule in the justification column, and list all of the assumptions inherited from

the premises in the assumptions column. Example 1 shows an instance of ⊃I. Here's another proof that shows examples of both ∨E and ∧E:

Example 2: ((P ⊃ Q) ∨ (Q ∧ R)), ¬(P ⊃ Q) ⊢$_{PL}$ R

1	(1)	((P ⊃ Q) ∨ (Q ∧ R))	Premise
2	(2)	¬(P ⊃ Q)	Premise
1, 2	(3)	(Q ∧ R)	1, 2, ∨E
1, 2	(4)	R	3, ∧E

Example 2 also illustrates a couple of other things. First, the letters in the inference rules are sentence variables, and can stand in for complex sentences as well as atomic sentences. On line (1), for example, ((P ⊃ Q) ∨ (Q ∧ R)) counts as an instance of P ∨ Q for the purposes of ∨E, because ∨E applies to all disjunctions (all sentences whose main connective is ∨). The inference rules tell you what to do with any sentence whose main connective is the one they're concerned with, no matter how complex the subsentences involved.

You may have noticed that there are two inference rules missing from the set above: ⊃I and ¬I. That's because these are both a bit more complex, and in similar ways. To prove a conditional (P ⊃ Q), we use the strategy of temporarily assuming its antecedent P, then using that assumption to "prove" Q, then taking that "proof" to demonstrate that if P, then Q (i.e, (P ⊃ Q)). I'll use the following notation to represent this:

⊃I [P]…Q ⊢$_{PL}$ (P ⊃ Q)
a.k.a. conditional-introduction, conditional proof

In this notation, the square brackets around the initial premise P indicates that it is a temporary assumption rather than a premise, and the ellipsis leading from P to Q indicates that P is temporarily assumed in order to prove Q, at which point this temporary assumption is discharged and we can draw (P ⊃ Q) as a conclusion. Temporary assumptions go in the assumption columns of proofs, but their temporary status is marked with square brackets, and they go away once ⊃I is invoked. The intuitive idea behind all of this is that ⊃I allows you to perform a sub-proof within your overall proof, thereby deriving the conditional. Here's an example:

Example 3: Q ⊢$_{PL}$ (P ⊃ (P ∧ Q))

1	(1)	Q	Premise
[2]	(2)	P	Assumption for ⊃I
1, [2]	(3)	(P ∧ Q)	1, 2, ∧I
1	(4)	(P ⊃ (P ∧ Q))	2, 3, ⊃I

By assuming P in line (2), we're able to prove (P ∧ Q) at line (3), and this is a proof of the conditional (P ⊃ (P ∧ Q)) at line (4), which is our conclusion, relying only on the premise Q from line (1).

We also rely on temporary assumptions and sub-proofs in ¬I, which is better known by its latin name, *reductio ad absurdum* (roughly: reduction to absurdity), because it involves making a temporary assumption and showing that that assumption leads to an absurdity—i.e., a contradiction, a sentence of the form (P∧¬P). I'll represent the rule like this:

¬I [P]…(Q∧¬Q) ⊢$_{PL}$ ¬P
a.k.a. negation-introduction, indirect proof, reductio ad absurdum, or just reductio

Here's a proof that uses ¬I. It also illustrates something else—the possibility of proving a conclusion from no premises. This is called proving a *theorem*. The theorem being proved here is called the Law of Non-Contradiction.

Example 4: ⊢$_{PL}$ ¬(P∧¬P)

[1]	(1)	¬¬(P∧¬P)	Assumption for ¬I
[1]	(2)	(P∧¬P)	1, ¬E
	(3)	¬(P∧¬P)	1, 2, ¬I

Here we start by temporarily assuming the negation of what we ultimately intend to prove, show that our temporary assumption leads to a contradiction—in this case, in a single step—and then draw our un-negated conclusion. Again, the intuitive idea is straightforward: if our temporary assumption leads to a contradiction, it must have been a bad assumption, and so we can conclude by inferring its negation. Reductio is one of the most commonly-used proof techniques in both philosophy and mathematics.

Without going into the details, I will say that the ten inference rules I've outlined so far add up to a complete set, in the sense that anything that we could want to prove in PL is provable with them. Of course, that's not to say that there aren't plenty of others; there are many other useful inference rules available, all of which can be derived from these ten by proving them. Once a proof of some sequent has been given, it can be treated as a new inference rule. And once a theorem has been proven, it can be stuck in as a new line of a proof for free (without writing any assumptions to the left). So, by starting with a small number of inference rules and no theorems, we can wind up with a lot of both, making our system more powerful as we go. The logic books that I recommended earlier contain lots of examples of other inference rules.

4.4.1 Soundness and Completeness

Here's an interesting question: do the semantic consequence relation defined using truth tables and the proof-theoretic consequence relation defined using natural deduction correspond to each other? Do they tell us that the same sentences of PL follow from one another? Are they just the same relation, defined in two different ways? Is $\models_{PL}$ the same as $\vdash_{PL}$? To ask these questions is to ask whether our proof theory is *sound* and *complete* with respect to our semantics (i.e., it is to ask whether $\vdash_{PL}$, as we've defined it, is sound and complete with respect to $\models_{PL}$, as we've defined it).

Soundness A proof-theoretic consequence relation $\vdash$ is sound with respect to a semantic consequence relation $\models$ if and only if, for every possible set of premises $\{P_1, P_2, \ldots P_n\}$ and every possible conclusion Q, if $\{P_1, P_2, \ldots P_n\} \vdash Q$, then $\{P_1, P_2, \ldots P_n\} \models Q$.

(In English: a proof theory is sound with respect to a semantics if and only if every instance of the proof-theoretic consequence relation in question is also an instance of the semantic consequence relation in question.)

Completeness A proof-theoretic consequence relation $\vdash$ is complete with respect to a semantic consequence relation $\models$ if and only if, for every possible set of premises $\{P_1, P_2, \ldots P_n\}$ and every possible conclusion Q, if $\{P_1, P_2, \ldots P_n\} \models Q$, then $\{P_1, P_2, \ldots P_n\} \vdash Q$.

(In English: a proof theory is complete with respect to a semantics if and only if every instance of the semantic consequence relation in question is also an instance of the proof-theoretic consequence relation in question.)

Without going into the details, I will say that $\vdash_{PL}$ is, in fact, sound and complete with respect to $\models_{PL}$. These are *metalogical* results—facts that can be proved logically about logics. If you take an intermediate logic course, you'll learn how to prove these and other metalogical results.

5 First-Order Logic

PL is a very simple language. This is nice because it makes the language, its syntax, its semantics, and its proof theory easy to learn. But one result of PL's simplicity is that it can't capture many intuitively good inferences. One example is the inference from 'Bertrand knows logic' to 'Someone knows logic'. In PL, these would both be atomic sentences—say P and Q—and so there's no inference rule or truth table that can tell us how to get from one to the other.

First-Order Logic (a.k.a. Predicate Logic) captures inferences like this one by discerning structure within atomic sentences. We'll call the language of First-Order Logic FOL (for 'First-Order Language). The syntax of this language is still much, much simpler than English, but it takes us one step closer. Like English, the atomic sentences of FOL are made up of *terms* and *predicates*. For example, in the English sentence 'Bertrand knows logic', we can distinguish the term (in this case, a proper name) 'Bertrand' and the predicate 'knows logic'. FOL represents predicates with uppercase letters—for example, 'knows logic' as K—and terms as lowercase letters—for example, 'Bertrand' as 'b'. In FOL sentences, terms are placed after predicates, so 'Bertrand knows logic' could be translated as Kb.

Some sentences of English have multiple terms in them, and the same is true of FOL. For example, the sentence 'Gottlob taught Bertrand' consists of two names, 'Gottlob' and 'Bertrand', and a two-place predicate, 'taught'. Sentences like these can be translated into FOL as Tgb, with T translating the two-place predicate 'taught', and b and g translating the names 'Gottlob' and 'Bertrand'. Some English sentences involve three-place predicates, four-place predicates, and so on. For example: 'New York is between Boston and Philly' could be translated as Bnbp. (When translating multiple-place predicates, it is important to keep track of what the order of the terms represents; in this example, the first term represents the thing that is between the things represented by the other terms.)

FOL also gains expressive power compared to PL in that it contains quantifiers. We have many quantifiers in English, including 'something', 'every Tuesday', 'a philosopher', and 'all dogs'. FOL contains only two quantifers: $\forall$ and $\exists$.[6] Quantifier symbols are used in sentences of FOL like the following ones. Here, the predicate F is used to translate the English predicate 'is fun'. The English sentence below each sentence of FOL are all equivalent ways of translating the sentence, starting with the one that most closely matches the FOL sentence's structure, and gradually moving toward more natural ways of saying the same thing.

(1) $(\forall x)Fx$
For every x, x is fun
Every x is fun
Everything is fun.

(2) $(\exists y)Fy$
There exists a y such that y is fun.
There exists a y that is fun.
There exists something that is fun.
Something is fun.

In these sentences, *x* and *y* are *variables*, which are terms, like names. Unlike names,

[6] Bertrand Russell was the first to use $\exists$ to annotate existential quantification—presumably because it was an easy way for the printer to typeset his manuscripts without creating new letterpress characters—but instead of $(\forall x)$ for universal quantification, he just wrote (x). The $\forall$ symbol was first introduced decades later, apparently by the logician Gerhard Gentzen in 1935 by analogy with $\exists$.

however, variables don't refer to anything unless they're temporarily assigned a referent. In FOL, they play the role of placeholders that can be *bound* by quantifiers. We write quantifiers in parentheses with variables next to them to indicate which variables they bind in what follows them. For example, $(\forall x)$ binds the variable x in (1). A sentence with an unbound variable in it—such as Fx by itself—is called an *open sentence*. Unless we temporarily assign a referent to x, Fx doesn't say anything. Intuitively, we can think of open sentences as standing for properties. For example, we can think of Fx as standing for the property of being F; if F translates 'is Fun', then, Fx can be thought of as the property of being fun. When we append a quantifier to the front of an open sentence in such a way that the quantifier binds the open sentence's variables, we create a sentence that generalizes in some way about which things have the property that the open sentence would stand for. For example, we can think of $(\forall x)$Fx as saying that everything has the property that Fx stands for—i.e., that everything is fun. And we can think of $(\exists y)$Fy as saying that there is at least one thing with the property that Fy stands for—i.e., that something is fun.

Because FOL contains all the same connectives as PL, we can build up a complex open sentence with several instances of the same variable in it, and then bind all of those instances with a single quantifier. For example, $(\exists x)(\text{F}x \wedge \text{G}x)$ says that at least one thing has the property that the open sentence $(\text{F}x \wedge \text{G}x)$ stands for—i.e., that something is both F and G.

Things can get somewhat more complicated after this, especially when we get to sentences containing more than one quantifier. For example, here are some other English sentences and their FOL translations—again, translating each name and predicate letter the same way throughout:

(3) Fa
Andy is fun.

(4) Gb
Bob is greedy.

(5) Pb
Bob is a person.

(6) Bba
Bob is bigger than Andy.

(7) F$a \supset$ Gb
If Andy is fun, then Bob is greedy.
Bob is greedy if Andy is fun.
Andy is fun only if Bob is greedy

(8) F$a \supset (\forall$x)Fx
If Andy is fun, then everything is fun.
Everything is fun if Andy is fun.

(9) $(\forall x)(\text{P}x \supset \text{G}x)$
For every x, if x is a person, then x is greedy.
Every person is greedy.
Everyone is greedy.

(10) $(\exists x)$Bbx
There exists an x such that Bob is bigger than x.
There is something such that Bob is bigger it.
Bob is bigger than something.

(11) $(\exists x)$Bxb
There exists an x such that x is bigger than Bob.
There is something such that is bigger than Bob.
Something is bigger than Bob.

(12) $(\exists x)(\forall y)$Bxy
There exists an x such that for every y, x is bigger than y.
There exists something such that it is bigger than everything.
Something is bigger than everything

(13) $(\forall y)(\exists x)$Bxy
For every y, there exists an x such that x is bigger than y.
Everything is such that something is bigger than it.
Something is bigger than everything.

Notice that the last two FOL examples can be paraphrased using the same sentence of English—'Something is bigger than everything'. This is not because the two FOL sentences mean the same thing (they don't); it's because the English sentence is ambiguous. It is common to call ambiguities of this kind *scope ambiguities*; they arise because natural language sentences containing more than one quantifier aren't always clear about which one has widest scope. Remember from our discussion of PL that the connective with widest scope is the main connective of a sentence. The same goes for quantifiers, and every well-formed sentence of FOL makes it clear which quantifier, if any, is the main one: if there is a quantifier all the way at the left of the sentence, and the parentheses to its right (and to the right of any other quantifiers to its right) enclose the rest of the sentence, then it is the main quantifier. In that case we say that the sentence is an existential quantification or a universal quantification (depending on the quantifier).

One reason that philosophers use the notation of First-Order Logic is so that they can avoid scope ambiguities and other kinds of structural unclarity that crop up in English

sentences. Another example follows, in which (15) and (16) are two possible translations of (14), and in which Dx translates 'x is a dog', Kx translates 'x is a cat', and Cxy translates 'x chases y':

(14) Every dog chases a cat.

(15) $(\forall x)(\exists y)((\mathrm{D}x \wedge \mathrm{K}y) \wedge \mathrm{C}xy)$
For every x, there is at least one y such that x is a dog, y is a cat, and x chases y.
For every dog, there is at least one cat that it chases.
Every dog chases a (potentially distinct) cat.

(16) $(\exists y)(\forall x)((\mathrm{D}x \wedge \mathrm{K}y) \wedge \mathrm{C}xy)$
There is at least one y such that for every x, x is a dog, y is a cat, and x chases y.
There exists a cat that is chased by every dog.

Another reason that philosophers use FOL is that it has the expressive power of both syllogistic logic and propositional logic, both at once. For example, you can use FOL to capture the Barbara inference as follows.

$(\forall x)(Fx \supset Gx)$
$(\forall x)(Gx \supset Hx)$
$(\forall x)(Fx \supset Hx)$

Notice, though, that the sentences here are no longer treated as having subject–predicate form. Rather, they are universally quantified conditionals. Modus Ponens also works:

$Fa \supset Gb$
Fa
Gb

So, FOL gives us the expressive power of both of the earlier formal languages combined, with the added bonus that it makes some claims expressibile that weren't expressible in either previous language. This is the main reason why it has become so universally used.

I want to introduce just one last piece of FOL's vocabulary, which is the identity symbol, =. In the context of FOL, = is a two-place relational predicate, but it is different than a normal relational predicate in two ways. First, it is a piece of logical vocabulary rather than nonlogical vocabulary; as we'll see, it gets its own semantic clauses and inference rules. Second, = has a different syntax than most predicate letters. Whereas all of the terms associated with a predicate are normally written after it, as in Cxy in (15) and (16), = is written between its two terms, as in $x = y$, which always translates 'x is identical to y'. By 'identical' here, we mean that x and y are the very same thing—that they are numerically identical, not merely that they are similar.

Because FOL is an artificial language that is very commonly used by philosophers, the ability to look at a sentence of FOL and figure out what it means (perhaps by translating it to yourself) is a perhaps the single most important kind of logic literacy required for doing analytic philosophy. The following textbook chapters review lots of examples, suggest some techniques, and contain practice exercises:

- *A Modern Formal Logic Primer* by Teller, Chs.3–4
- *An Introduction to Formal Logic* by Smith, chs.21–24, pp.194–227

I turn now to the syntax, semantics, and proof-theory for FOL, which I will treat in less detail than I did for PL.

5.1 Syntax of FOL

We start with the basic vocabulary:

Individual constants (names): a, b, c,…
(It is customary to use lowercase letters from the beginning of the alphabet for these.)

Variables: $x, y, z, \ldots x_1, x_2, \ldots$
(It is customary to use lowercase letters from the end of the alphabet for these.)

(A *term* is anything that is either an individual constant or a variable.)

Predicate Constants: A, B, C,…F,…P…
(Each predicate constant has to have a specific number of argument places, which is the number of terms that come after it to make a sentence. A 2-place constant letter takes two terms, as in Fxy. An n-place predicate letter takes n terms. In principle, n can be as big as we want. In some logic textbooks, the number of argument places a predicate has is represented by subscripts (F_3abc) or superscripts (F^2yz), but we'll omit those here.)

The identity predicate: =

The Sentence Connectives: $\neg, \wedge, \vee, \supset, \equiv$
(These are the same connectives from PL, and the same alternative notations mentioned above in Table 1 are also commonly found in FOL.)

Quantifier Symbols: $\forall, \exists$

Parentheses: (,)

Next we define what it takes to be a sentence of FOL. This goes in two steps. First we say what is to be a *formula* of FOL:

(I) If F is an n-place predicate letter and $t_1, t_2, \ldots t_n$ are n terms, then $\mathrm{F}t_1t_2 \ldots t_n$ is a formula of FOL.

(II) If t_1 and t_2 are terms, then $t_1 = t_2$ is a formula of FOL.

(III) For any two formulae of FOL, φ and ψ, the following are also formulae of FOL:

(i) $\neg\varphi$

(ii) $(\varphi \wedge \psi)$

(iii) $(\varphi \vee \psi)$

(iv) $(\varphi \supset \psi)$

(v) $(\varphi \equiv \psi)$

(IV) If x is a variable and φ is a formula of FOL, then the following are also formulae of FOL:

(i) $(\exists x)\varphi$

(ii) $(\forall x)\varphi$

(V) No other string of symbols is a formula of FOL.

Next, with the help of precise definitions of scope and variable binding, we can define what it is to be a sentence of FOL:

Scope: If φ is a formula of FOL, then: (a) in every formula of FOL of the form $(\forall x)\varphi$, φ is the scope of $(\forall x)$, and (b) in every formula of FOL of the form $(\exists x)\varphi$, φ is the scope of $(\exists x)$.

Bound vs. Free Variables: A variable x is bound by a quantifier $(\forall x)$ (or $\exists x$) in a formula φ if and only if (a) x is contained in φ; (b) φ is the scope of $(\forall x)$ (or $\exists x$); and (c) x is not in the scope of any other quantifers of the form $(\forall x)$ or $(\exists x)$ within φ. Any variable that is not bound in a formula φ is free in φ.

Sentences A sentence of FOL is a formula of FOL with no free variables.

All of this just formalizes and precisifies what I've already said. We need to proceed in two steps in order to avoid having sentences containing free variables—open sentences, in the terminology I used earlier. As I said earlier, they don't have meanings on their own—not unless we temporarily assign referents to the variables.

Before proceeding to semantics, I want to mention a couple of kinds of sentences of FOL that may seem weird. For example, some sentences of FOL contain quantifiers that don't do anything—for example, $(\exists x)\mathrm{F}a$, which could be used to translate 'There is something such that Albert is fat'. In this English sentence, the quantifier 'there is something such that' doesn't add anything to the sentence; it is a mere circumlocution. The same is true of the quantifier in the FOL sentence just mentioned, and this is because that quantifier doesn't bind any variables. Intuitively, an existential quantifier $(\exists x)$ can be thought of as saying that whatever is predicated of free occurrences of x in its scope is true of at least one thing. If there are no free occurrences of x in its scope, then the quantifier adds nothing to the sentence. But it's still a perfectly well-formed sentence, just not a very useful one—much like the English sentence just mentioned.

Other weird sentences involve multiple occurrences of the same quantifier, or two quantifiers that bind the same type of variable. These sentences can be confusing, because it can be difficult to tell what's binding what in them. But our syntax—particularly, clause (b) of our definition of bound variables—gives us a way of understanding such sentences, and there is always a way to switch out some of the variables in them to make them less confusing. Here are a couple of examples, together with clearer equivalent sentences:

(17) $(\exists x)(\mathrm{F}x \wedge (\forall x)\neg\mathrm{F}x)$
$(\exists x)(\mathrm{F}x \wedge (\forall y)\neg\mathrm{F}y)$

There is at least one x such which is F and for every x, x is not F.

Something is F and everything is not F.

(18) $(\forall x)(\forall y)(\mathrm{R}xy \supset (\exists y)(\mathrm{R}xy))$
$(\forall x)(\forall y)(\mathrm{R}xy \supset (\exists z)(\mathrm{R}xz))$

For every x and every y, if x bears R to y, then there is something to which x bears R.

For any two things, if one is related to another by R, then there is something to which it is related.

The trick with these sentences is to see that a variable is always bound by the matching quantifier with the smallest scope that includes the variable.

5.2 Interlude: A Bit of Set Theory

Set theory is a branch of mathematics that studies sets—collections of things considered as wholes. Set theorists study things like the foundations of mathematics and the properties of different kinds of infinite numbers.

We need to know a bit of set-theoretic notation for a couple of reasons. First, this is one of the kinds of notation that pops up from time to time in philosophy papers, including in the philosophy of language. Second, we need to know a bit of set theory because the best-known semantics for First-Order Logic is articulated using set theory.

So, A *set* is just a collection of things considered as a whole. If A is a set and B is a set, then A and B are identical (A = B) if and only if they have all and only the same members. We can define sets in a couple of ways:

- {a, b, c}
  This is the set containing a, b, and c. To define a set by listing its members in this way is to define it *in extension*.
- $\{x : Fx\}$
  This is the set of all x's such that x is F—i.e., the set of everything that meets the condition F. We can put any open sentence with x free to the right of the colon to specify some condition for the set to meet. This is a way of specifying a set without naming each of its members, which is useful for very large (including infinitely large) sets. To define a set by specifying a property shared by all of its members in this way is to define the set *in intension*.

Here are some standard notations for talking about relationships that hold between sets and their members:

Membership $a \in A$
a is an element of the set A
(alternatively: a is a member of the set A)

Subset: $A \subseteq B$
A is a subset of B
(This is true if and only if every member of A is also a member of B. This is compatible with the possibility that A=B. Every set is a subset of itself.)

Proper Subset: $A \subset B$
A is a proper subset of B
(This is true if and only if every member of A is also a member of B and every member of B is not also a member of A. In symbols: $A \subset B$ iff $A \subseteq B$ and $B \not\subseteq A$.)

Superset: $A \supseteq B$
A is a superset of B
(This is true if and only if every member of B is also a member of A. This is compatible with the possibility that A=B. Every set is a superset of itself. For any two sets, $A \subseteq B$ if and only if $B \supseteq A$.)

Proper Superset: $A \supset B$[7]
A is a proper superset of B
(This is true if and only if every member of B is also a member of A and every member of A is not also a member of B. In symbols: $A \supset B$ if and only if $A \supseteq B$ and $B \not\supseteq A$.)

We can also use some standard notation to define sets in terms of other sets. Here A and B stand for arbitrary sets:

Union: $A \cup B = \{x : x \in A \vee x \in B\}$
The *union* of two sets A and B is the set containing everything that is a member of either A or B.

Intersection: $A \cap B = \{x : x \in A \wedge x \in B\}$
The *intersection* of two sets A and B is the set containing only those things that are members of both A and B.

Complement: $A—B = \{x : x \in A \wedge x \notin B\}$(alternative symbol: $A \backslash B$)
The *complement* of a set A in a set B is the set containing all the members of A that are not also members of B.

Powerset: $\mathbb{P}(A) = \{X : X \subseteq A\}$
The *powerset* of a set A is the set of all of the subsets of A.

There are also a few special kinds of sets that we need to be able to talk about.

Ordered Tuples Normally, the things in a set have no particular order. {x, y} is the same set as {y, x}. An ordered tuple is a special kind of set whose members' order matters. We write ordered tuples with angled brackets, as follows:

- $\langle x, y \rangle$
  The ordered pair consisting of x followed by y.
- $\langle a, b, c \rangle$
  The ordered triple consisting of a followed by b followed by c.
- $\langle x_1, x_2, \ldots x_n \rangle$
  The ordered n-tuple consisting of n elements, $x_1, x_2, \ldots x_n$. (Here, n stands in for any natural number. An n-tuple is a tuple with n ordered elements.)

Relations Intuitively, a relation is like a property, except that it connects more than one thing together. For example, the relation of *being greater than* connects each number to each other number that is smaller than it.

A relation can be represented by a set of tuples. A binary relation is represented by a set of ordered pairs, each one representing a pair of things that are so-related. For example, the relation of *being greater than* can be represented by a set of all the

[7] Notice that the symbol for 'is a proper supserset of' is the same one used for 'if…then'. This is annoying coincidence, but it shouldn't be too confusing if you keep track of when we're talking about sentences and when we're talking about sets.

pairs of numbers such that the first member of the pair is greater than the second: $\{\langle 2,1\rangle, \langle 3,1\rangle, \langle 3,2\rangle, \ldots\}$. (Since there are infinite numbers, this will be an infinitely large set.) The relation of being a number between two other numbers, which is a ternary (three-place) relation (x is between y and z) can be represented by an infinitely large set of ordered triples: $\{\langle 2,1,3\rangle, \langle 3,2,4\rangle, \langle 4,3,5\rangle \ldots\}$.

Functions Intuitively, we can think of a function as a rule or process that takes one thing as an input and outputs another thing. What makes functions useful is that they always yield the same output for every input. We can also call their inputs *arguments* and their output for any given argument their *value* for that argument. Another way of saying all this is to say that functions are mappings from their inputs (arguments) to their outputs (values).

We can represent a function as a special kind of binary relation—as a set of ordered pairs of a special kind. Functions are relations that never relate something in their domain to more than one thing in their range. For each of the ordered pairs that makes up a function, $\langle x, y\rangle$, x is an argument of the function and y is the value of the function for that argument. The set of all of a function's arguments are its *domain* and the set of all of its values is its *range* (a.k.a. *image*).

For any function f, $f(x)$ is the function's value (output) given x as an argument (input). An example would be the squaring function, which we could symbolize as square( ) (e.g., square(2) = 4), or as $(\,)^2$ (e.g. $(2)^2 = 4$). (This latter notation is standard in mathematics, but without the parentheses.)

Cartesian Products $\mathrm{A}\times\mathrm{B} = \{\langle x, y\rangle : x \in A \wedge y \in B\}$

The Cartesian product of two sets A and B is the set of ordered pairs such that the first member of each pair is a member of A and the second member of each pair is a member of B. (I.e., the product of A and B is the set of pairs of A-members with B-members.)

We can write a three-way product like this: $\mathrm{A}\times\mathrm{B}\times\mathrm{C}$. This is the set of ordered triples whose first member is a member of A, second member is a member of B, and third member is a member of C. So on with four-way products.

It is often useful to talk about the n-way product of a set with itself, which we write A^n. For example: $\mathrm{A}\times\mathrm{A} = \mathrm{A}^2$, $\mathrm{A}\times\mathrm{A}\times\mathrm{A} = \mathrm{A}^3$, and so on.

5.3 Semantics for FOL

The semantics for FOL works differently than the semantics for PL. The intuitive idea is that we'll show how the truth value of each sentence of FOL depends systematically on the meanings of the sentence's parts. Using this, we can show how the truth values of different sentences of FOL depend on one another, which will allow us to define a semantic consequence relation for FOL that meets the same definitions from earlier.

We start with a set, $\mathbb{U}$—the *universe of discourse*—which contains all of the entities that the sentences of FOL can be about. Intuitively, this is just the set of all of the things in the world that we can talk about. Next, we show how to define an *interpretation function* that maps each item of FOL's non-logical vocabulary to an element of $\mathbb{U}$ or a set built up from the members of $\mathbb{U}$. Intuitively, an interpretation function maps each expression to its meaning—also known as its *denotation*, *extension*, or *referent*. It is common to annotate the interpretation function using double square brackets, $[\![e]\!] = d$, with e standing in for an expression and d standing in for the expression's denotation. First, we define the denotation of each piece of non-logical vocabulary:

(I) For every name a, $[\![a]\!] \in \mathbb{U}$
The denotation of a name is an entity in the universe of discourse.
(Intuitively, a name's denotation is the thing to which the name refers.)

(II) For every 1-place predicate letter F, $[\![\mathrm{F}]\!] \subseteq \mathbb{U}$
The denotation of a 1-place predicate is a set of elements in the universe of discourse.
(The intuitive idea here is that the denotation of a predicate is a property, which we model as the set of things in the world of which the predicate is true. For example, the denotation of 'red' is the set of things that are red.)

(III) For every n-place predicate letter G such that n is greater than 1, $[\![\mathrm{G}]\!] \subseteq \mathbb{U}^n$
The denotation of an n-place predicate is an n-ary relation on the universe of discourse—i.e., a set of ordered n-tuples of elements in the universe of discourse.
(The intuitive idea here is that the denotation of an n-place predicate is an n-ary relation, which we model as the set of n-tuples of things in the universe of discourse of which the predicate is true. For example, the denotation of 'greater than' is the set of pairs such that the first member of the pair is greater than the second member of the pair.)

We also need to define the interpretation function for variables. Since they don't have meanings on their own, we do this by first defining denotation of a variable relative to a temporary *variable assignment*. We define a variable assignment as follows:

Variable Assignment: For any variable assignment f and variable x, $f(x) \in \mathbb{U}$
A variable assignment f is a function that maps each variable to an entity in the universe of discourse.

Next we define the denotation of a variable relative to an assignment:

(IV) For every variable x, $[\![x]\!]_f = f(x)$
The denotation of a variable x relative to a variable assignment f is whichever entity f assigns to x.

As we'll see, this variable assignment stuff is just temporary, in the sense that variable assignments play no role in the final definitions of semantic consequence. They're just a stepping stone to make things simpler on the way there.

The next steps tell us what it is for sentences of FOL to be true. We do this by further extending our definition of the interpretation function, which defines the conditions under which formulae of FOL are true.

(V) For any n-place predicate F and any n terms $t_1, t_2, \ldots t_n$, $[\![\mathrm{F}t_1, t_2, \ldots t_n]\!]_f$ = true if and only if $\langle [\![t_1]\!]_f, [\![t_2]\!]_f \ldots [\![t_n]\!]_f \rangle \in [\![F]\!]$.

An atomic sentence of FOL is true if and only if the ordered tuple of the denotations of its terms is a member of the denotation of its predicate. For example: Fa is true if and only if the denotation of a is in the set which is the denotation of F.

(VI) For any terms, t_1 and t_2, $[\![t_1 = t_2]\!]_f$ is true if and only if $[\![t_1]\!]_f$ is the same member of $\mathbb{U}$ as $[\![t_2]\!]_f$.

(VII) For any two formulae of FOL, φ and ψ:

(i) $[\![\neg\varphi]\!]$ is true if and only if $[\![\varphi]\!]$ is not true.

(ii) $[\![\varphi \wedge \psi]\!]$ is true if and only if both $[\![\varphi]\!]$ is true and $[\![\varphi]\!]$ is true.

(iii) $[\![\varphi \vee \psi]\!]$ is true if and only if either $[\![\varphi]\!]$ is true or $[\![\varphi]\!]$ is true.

(iv) $[\![\varphi \supset \psi]\!]$ is true if and only if either $[\![\varphi]\!]$ is not true or $[\![\varphi]\!]$ is true.

(v) $[\![\varphi \equiv \psi]\!]$ is true if and only if both $[\![\varphi]\!]$ and $[\![\varphi]\!]$ have the same truth value.

(These statements are equivalent to the truth tables for the connectives given for PL.)

(VIII) For any FOL formula φ and variable x, $[\![(\forall x)\varphi]\!]_f$ is true if and only if, for every variable assignment g which differs from f at most in what it assigns to x, $[\![\varphi]\!]_g$ is true.

The intuitive idea here is that the truth of a universally quantified formula is guaranteed when the formula is true no matter what is assigned to the variable bound by the universal quantifier. So, for example: $(\forall x)\mathrm{F}x$ is true if and only if Fx is true no matter what we assign to x. In other words: $(\forall x)\mathrm{F}x$ says that every object in the domain of discourse has the property picked out by Fx.

(IX) For any FOL formula φ and variable x, $[\![(\exists x)\varphi]\!]_f$ is true if and only if, for at least one variable assignment g which differs from f at most in what it assigns to x, $[\![\varphi]\!]_g$ is true.

This is similar to the clause for universal quantification, but the truth of $(\exists x)\mathrm{F}x$ requires only that Fx be true on at least one way of assigning a denotation to x. In other words: $(\exists x)\mathrm{F}x$ says that at least one thing in the domain of discourse has the property picked out by Fx.

(X) For any FOL sentence φ, $[\![\varphi]\!]$ is true if and only if $[\![\varphi]\!]_f$ is true for any variable assignment f.

Variable assignments are just temporary ways of giving meanings to free variables. Since a sentence is a formula with no free variables, we remove all reference to variable assignments at this stage and define truth for sentences in a way that is no longer relativized to variable assignments. (This is what makes variable assignments *temporary*: they're only needed during the intermediate stages of the semantics, and are discarded at the end.)

Clauses (I)–(X) together give a complete definition of what it is for a sentence of FOL to be true. If we know what an interpretation function assigns to each item of non-logical vocabulary, then we can mechanically find out whether any sentence is true or false. But of course, there are many different possible interpretation functions—many ways of mapping the non-logial vocabulary of FOL to their meanings. We take advantage of this fact in defining semantic consequence for FOL:

Semantic Consequence for FOL For any n FOL sentences $\varphi_1, \varphi_2, \ldots \varphi_n$ (the premises) and any FOL sentence, ψ (the conclusion), $\varphi_1, \varphi_2, \ldots \varphi_n \models_{FOL} \psi$ if and only if every interpretation function which makes each of $\varphi_1, \varphi_2, \ldots \varphi_n$ true also makes ψ true.

This says that a conclusion follows from a collection of premises just in case any interpretation of the non-logical vocabulary that makes the premises true also makes the conclusion true—which is the same thing as saying that there is no way of interpreting the premises to make them true which doesn't also make the conclusion true.

(Just as we used $\models_{PL}$ to symbolize the semantic consequence relation for PL, we use $\models_{FOL}$ to symbolize the semantic consequence relation for FOL.)

Recall, again, the three ways of defining semantic consequence from earlier:

(i) A sentence B follows from a set of sentences {X,Y,Z…} if and only if the truth of all of X,Y,Z…would guarantee the truth of B.

(ii) A sentence B follows from a set of sentences {X,Y,Z…} if and only if B is true in every possible situation in which all of X,Y,Z…are true.

(iii) A sentence B follows from a set of sentences {X,Y,Z…} if and only if, no matter what the non-logical vocabulary in X,Y,Z…and B mean, if all of X,Y,Z…are true then B is true.

Clearly, our definition of $\models_{FOL}$ is tailor-made to meet the third definition. But we can also think of each possible interpretation function as representing a possible situation of the world. In one possible situation, a given predicate (say, 'is bald') will denote one set of things (say, Michael Jordon, Dr. Evil, etc., because those are the people who are bald in that situation), and in another possible situation, the same predicate will denote different things (e.g., Martha Stewart, who might have been bald in some other possible situation). Our definition of $\models_{FOL}$ says that a conclusion follows from some premises just in case there is no possible situaton that could make the premises true and the conclusion false. By the same token, there is an obvious sense in which our definition of consequence captures the idea of the truth of some premises guaranteeing the truth of a conclusion.

Our definition of semantic consequence for PL was quite practical, in that it gave us a simple and reliable way of checking the validity of any argument by creating a truth table for it. Our definition of semantic consequence for FOL is harder to use for checking arguments' validity, but it is no less precise, and can often be used to show that an argument is not valid, by building an example of an interpretation function that makes its premises true and its conclusion false. For example, take the argument $\mathrm{F}a \models_{FOL} (\forall x)\mathrm{F}x$. We can show that this is an invalid argument by building the following counter-interpretation:

- $\{y, z\} \subseteq \mathbb{U}$

  The universe of discourse contains (at least) the entities y and z.

- $[\![a]\!] = y$

  The name a denotes the entity y

- $[\![\mathrm{F}]\!] = \{y\}$

  The predicate F denotes the set which contains only y—i.e., y is the only entity of which F can be truly predicated.

This interpretation makes Fa true, since the denotation of a is a member of the denotation of F. But it makes $(\forall x)\mathrm{F}x$ false, because at least one possible variable assignment would assign the variable x to the entity z, which is not a member of F's denotation—i.e., because the predicate F is not true of every entity. This shows that there is at least one interpretation function that makes the premise of our argument true and the conclusion false, which shows that the argument does not live up to our definition of semantic consequence.

For fuller treatments of semantics for first-order logic, have a look at the following book chapters:

- *An Introduction to Formal Logic* by Peter Smith, chs.27–28
- *The Logic Book* by Merrie Bergmann, James Moor, & Jack Nelson, Ch.8

5.4 Proof-Theory for FOL

(Treat this section as optional.) I won't dwell for too long on the proof-theory for FOL, because it is basically an extension of the proof-theory for PL: all of the inference rules that work for PL also work for FOL (but with FOL sentences substituted for PL sentences of the same form). FOL's proof theory differs in that we need a few new rules for dealing with the quantifiers and with the identity predicate.

Before we can define those rules, we need one more piece of notation:

Substitution: $\varphi(a/x)$ is the formula we get from φ by replacing every free occurrence of the variable x in φ with the name a.

Here are our new inference rules:

∀E $(\forall x)\varphi \vdash_{FOL} \varphi(a/x)$

(A.k.a. Universal Elimination, Universal Instantiation, Universal Specification)

This says that if we have a universally quantified sentence on a line, we can write an instantiation of that sentence on a new line. An instantiation of a universally quantified sentence is like the original, but with the universal quantifier removed and each of the variables that had been bound by it replaced by occurrences of the same name.

The intuitive idea is that if everything has some property, then we can infer, about any particular thing, that it has the property too.

For example: from $(\forall x)(\exists y)(\mathrm{R}xy \wedge Fx)$ we can infer $(\exists y)(\mathrm{R}ay \wedge Fa)$.

∀I $\varphi(a/x) \vdash_{FOL} (\forall x)\varphi$ (Can be used only if (i) a does not occur in any undischarged assumption line, and (ii) a does not occur in $(\forall x)\varphi$.)

(a.k.a. Universal Introduction, Universal Generalization)

This says that if we have a sentence that predicates something of a name, and we haven't made any assumptions about the bearer of that name so far, we can infer that the property in question is true of everything. The intuitive idea here is that if we haven't made any assumptions about a thing, we can treat it as a completely arbitrary thing.

For example, if we've previously inferred Fa from $(\forall x)\mathrm{F}x$ via ∀E, and we haven't said anything else about a in our proof, we can now infer $(\forall x)\mathrm{F}x$ once again. In this case, a has been treated as a completely arbitrary object.

∃I $\varphi(a/x) \vdash_{FOL} (\exists x)\varphi$
(a.k.a. Existential Introduction, Existential Generalization)
This says that if we have any sentence with one or more occurrences of a name in it, we can infer a sentence like that one, but with any number of those occurrences of the name replaced with a variable, and with an existential quantifier that binds the variable appended to the start of the sentence.
The intuitive idea here is that if we know, of any particular thing, that it has a certain property, then we also know that there exists at least one thing that has the property.
For example: from Fa, we can infer $(\exists x)$Fx.

∃E $(\exists x)\varphi, [\varphi(a/x)] \ldots \psi \vdash_{FOL} \psi$ (Can be used only if (i) a does not occur in any un-discharged assumption, (ii) a does not occur in $(\exists x)\varphi$, and (iii) a does not occur in ψ.)
(a.k.a. Existential Elimination, Existential Instantiation)
This is the most complicated of our rules. It says that if we begin with an existentially quantified sentence $(\exists x)\varphi$ and we're able to prove some sentence ψ by temporarily assuming an appropriate substitution instance $\varphi(a/x)$ of the existential quantification, then we can conclude ψ.
The intuitive idea here is that $(\exists x)\varphi$ says that some thing or other meets whatever conditions φ places on it. So if we suppose the truth of that substitution instance, treating the name we substitute for x as a completely arbitrary one, we can trust whichever conclusions result.

These rules are explained in great detail, with practice exercises, in the following book chapter:

- *The Logic Book* by Merrie Bergmann, James Moor, & Jack Nelson, Ch.10

Although it can be helpful in reading philosophy to know how to recognize provable sequents in FOL, it is far less of an essential skill than the ability to translate back and forth between FOL and English.

6 Abbreviations

6.1 Truth-Functional Connectives

I have proceeded as if there is just the one unary truth functional connective ($\neg$) and four binary truth-functional connectives ($\wedge$, $\vee$, $\supset$, and $\equiv$). But some of these can actually be defined in terms of the others. For example, here is a way of reducing $\equiv$ to $\supset$ and $\wedge$:

Biconditional Abbreviation For any two sentences, φ and ψ:
$(\varphi \equiv \psi) =_{df} ((\varphi \supset \psi) \wedge (\psi \supset \varphi))$

Here, the $=_{df}$ symbol indicates that the stuff to the right is equivalent by definition of the stuff on the right. In other words, the left side is an abbreviation of the right side. This shows that we don't need to understand $\equiv$ as one of our basic symbols; we can think of sentences containing it as abbreviations for more complex sentences.

In fact, there are several ways to get by with only two connectives. Frege took $\neg$ and $\supset$ to be basic, and defined the rest in terms of them. Russell and Whitehead did the same in the first edition of *Principia Mathematica*. Here's how to do that:

Conjunction Abbreviation For any two sentences, φ and ψ:
$(\varphi \wedge \psi) =_{df} \neg(\varphi \supset \neg\psi)$

Disjunction Abbreviation For any two sentences, φ and ψ:
$(\varphi \vee \psi) =_{df} (\neg\varphi \supset \psi)$

It can easily be seen that these abbreviations are equivalent by building truth tables for them. In each of the cases listed above, the truth table for the left side will match the truth table for the right side.

This shows that anything we can say using $\neg$, $\wedge$, $\vee$, $\supset$, and $\equiv$ can likewise be said using just $\neg$ and $\supset$. To say this is to say that these two connectives are, together, functionally complete: they can be used to express any possible truth function. Some other combinations are likewise truth-functionally complete: $\neg$ and $\vee$ would be enough on their own, for example. In fact, it is possible to get by with just a single truth-functional connective. There are two options, sometimes called the Sheffer Stroke (a.k.a. NAND) and NOR, respectively.

P	Q	P ↑ Q
T	T	F
T	F	T
F	T	T
F	F	T

(a) Sheffer Stroke

P	Q	P ↓ Q
T	T	F
T	F	F
F	T	F
F	F	T

(b) NOR

Table 7: Truth Tables for Binary Connectives

Each of these connectives is functionally complete on its own: they can be used to abbreviate all of the usual connectives. This is exactly how Russell and Whitehead used the Sheffer Stroke in the Second Edition of *Principia Mathematica*, in order to reduce their primitive logical vocabulary to a bare minimum. For example, $\neg\varphi$ can be thought

of as an abbreviation of $(\varphi \uparrow \varphi)$, and $(\varphi \supset \psi)$ can be treated as an abbreviation of $(\varphi \uparrow (\psi \uparrow \psi))$.

Finally, notice that we can translate any sentence involving a universal quantifier into a sentence involving an existential quantifier instead. This means that we can get by with just one quantifier, or else that we can take one quantifier as primitive and treat one quantifier as an a device for abbreviation. In fact, Frege did the former and Russell did the latter. They both took the universal quantifier as a primitive. Here's a way for defining the existential quantifier away:

Existential Quantifier Abbreviation For any formula φ,
$(\exists x)\varphi =_{df} \neg(\forall x)\neg\varphi$

The other option would work just as well:

Universal Quantifier Abbreviation For any formula φ,
$(\forall x)\varphi =_{df} \neg(\exists x)\neg\varphi$

In general, you can always take any quantifier and replace it with the other quantifier, flanked by negations on each side, and wind up with a logically equivalent sentence. So we could choose to have either quantifier as part of our primitive vocabulary and treat the other one as defined.

What's the point of all this abbreviation? One thing is that it allows us to better understand the relationships between the different vocabulary we're working with. But Frege, Russell, and Whitehead were also interested in something else. They wanted to show that mathematics could be reduced to the bare minimum of logical concepts. So, reducing their primitive vocabulary as much as possible made sense for them as an exercise in conceptual frugality.

Moreover, this sort of abbreviation formed the template for a good amount of analytic philosophy that came after it. Here's an interesting epistemological question: what are the logically most basic concepts in terms of which our knowledge of the world could be framed. It seems unlikely that 'bachelor' is among these concepts, since we could just go around thinking of bachelors as unmarried men instead. But how far could this sort of analysis be pushed before we reach the most basic concepts that can't be further analyzed without missing anything? This is part of Russell's project in epistemology, and many of those after him have tried to do something similar.

6.2 Russell's Theory of Definite Descriptions

This brings us to a kind of abbbreviation that Russell and many after him took to be extremely significant. One way of understanding the issue is to ask how sentences like the following should be translated into FOL:

> The discoverer of the quantifier is smart.

Here, the expression 'the discoverer of the quantifier' is a *definite description*. On one hand, definite descriptions seem to work like proper names. They can be put into a sentence wherever a proper name can go. On the other hand, it seems that definite descriptions shouldn't be treated like proper names because they have internal structure that we might want our logic to be sensitive to. For example, the following inference seems valid:

> The discoverer of the quantifier is smart.
> Therefore, someone discovered the quantifier.

But, if we translate 'the discoverer of the quantifier' into FOL as a proper name (say, '*i*'), we can't capture the validity of this inference, since the thing about someone inventing the quantifier won't be part of our FOL sentence's structure. The predicate 'discovered the quantifer' won't be treated as an independently meaningful part of the sentence, much like 'trand' in 'Bertrand is British'. This seems wrong. As you'll see when you read 'On Denoting', there are many other puzzles connected to definite descriptions, so that treating them as something other than proper names is warranted.

Russell's response to these puzzles was to treat sentences containing definite descriptions as abbreviations for longer sentences. First, let's see his notation for writing definite descriptions in FOL. He would translate the this sentence of English

> The queen of England is short.

as this sentence of FOL (here, 'S' translates 'is short' and 'Q' translates 'is Queen of England':

$$S[(\imath x)Qx]$$

Here, the expression $[(\imath x)Qx]$ translates the definite description 'the Queen of England'. Russell treats expressions of this kind grammatically as terms. They can go wherever a name would go in a sentence of FOL. So, in this case, for example, the description goes after the predicate *S*, just where the name *a* would go in the FOL sentence *Sa*. A more direct translation of the above sentence would be 'The *x* such that *x* is Queen of England is short'.

The added complexity in $[(\imath x)Qx]$ allows sentences containing it to be de-abbreviated. Here is how Russell proposes to do that:[8]

[8] This is a slightly simplified version of Russell and Whitehead's equivalent definition, which they give as *14.01 of *Principia Mathematica*.

Definite Description Abbreviation For any predicates F and G,

$$G[(\imath x)Fx] =_{df} (\exists x)(Fx \wedge ((\forall y)(Fy \supset y = x) \wedge Gx))$$

Here's a literal translation into English:

> 'The F is G' $=_{df}$ 'There is at least one x such that: x is F, for any y, if y is F, then y is identical to x, and x is also G.'

Here's a more natural translation:

> 'The F is G' $=_{df}$ 'There is exactly one thing, x, which is F, and x is also G.'

And here's a rougher but even more natural translation:

> 'The F is G' $=_{df}$ 'At least one thing and at most one thing is F, and it is also G.'

A crucial thing to notice about this definition is that definite descriptions—expressions of the form $[(\imath x)Fx]$—turn out not to be terms after all. Once the abbreviation is taken away, there is no single expression that is itself the translation of $[(\imath x)Fx]$. For this reason, Russell called them "incomplete symbols"—expressions that look like unified expressions that stand for something on the surface, but that disappear when we analyze the sentence, removing its abbreviations, and revealing its true *logical form*. Once the true logical form of a sentence containing a description is revealed, there is no longer any expression that corresponds to the description.

Russell was excited about this theory for several reasons. One is that, showing how mathematics reduces to logic, he has to give a lot of definitions of things. A definition of a thing typically takes the form, '2 is the such-and-such'. In other words, they tend to contain a definite description that gives the identity of the thing being defined. With his theory of definite descriptions, he had a way of interpreting definitions of this form. The theory of descriptions was therefore Russell's crucial analytical tool.

Secondly, Russell thought that definite descriptions in natural language give rise to all sorts of paradoxes, and he thought that these paradoxes could be solved if we apply his theory of descriptions to natural language as well. He makes this proposal in 'On Denoting' and in some other places.

Third, Russell eventually came to think that the theory of descriptions was the key to his metaphysics and epistemology. Here's the problem: we seem to be capable of thinking and speaking about entities in the external world that we've never directly experienced. However, since we've never directly experienced them, it's unclear how our mind could make contact with them, such that we could have thoughts about them. Russell's solution is to say that we have knowledge of such entities *by description*. So, for example, here's something I take myself to know:

> Michael Jordan played basketball.

But I've never directly experienced Michael Jordan, so how is it possible for me to have a thought with him as its content? Russell's answer is that 'Michael Jordan' is not a logically proper name, but an abbreviated definite description. So we can get closer to the logical form of the sentence as follows:

> The x such that x won six NBA championship MVP awards in the 1990s (etc.) played basketball.

And this further translates, by Russell's theory of descriptions, as this:

> There is an x such that x won six NBA championship MVP awards in the 1990s (etc), x was the only one to do this, and x played basketball.

So now there's no expression in here that seems like it refers to Michael Jordan, and so we no longer have to explain how *he* could be one of the contents of my thoughts, despite my never having directly experienced him. My mind makes contact with Michael Jordan only indirectly, by knowing some stuff about him. This is what it is to have knowledge by description.

This is a lot like Russell's strategy for the epistemology of mathematics: explain our knowledge of mysterious-seeming entities (numbers, Michael Jordan) by translating sentences about those things into sentences that no longer seem to be about those things. This is the kind of analysis that originally gave analytic philosophy its name.

(This has just been a preview. We'll spend more time on Russell's theory of descriptions later in the course.)

7 Second-Order Logic

When people talk about 'Classical Logic', they usually mean First-Order Logic with Identity—i.e., the logic that I described in §5.[9] But the logics used by Frege, Russell, and Whitehead weren't actually first-order logic but *second-order* logic. What does this mean?

I won't go into all the details, but the intuitive idea is pretty simple. In first-order logic, quantifiers can only bind variables in term position, as in:

$$(\forall x)(Fx)$$

In second-order logic, there are also predicate variables, and these variables can be bound by second-order quantifiers. E.g.:

[9] I agree: it *is* confusing that this logic is called 'classical', given that it was discovered by Frege in 1879! I honestly don't know why people use this terminology.

$$(\forall \Phi)(\Phi a)$$

This could be translated as, 'for any property Φ, a has Φ'.

Second-order logic was important to Frege and Russell for a variety of reasons. One is that it's possible to translate sentences about sets into sentences of second-order logic. According to Russell at least, this is an important part of reducing mathematics to logic. Another is that the most obvious way of translating the last of Peano's axioms for arithmetic into logical notation uses second-order logic. Here's that postulate stated in English, by Russell in the introduction to mathematical philosophy:

> Any property which belongs to 0, and also to the successor of every number which has the property, belongs to all numbers.

And here is the most natural way of translating this into logical notation.

$$(\forall \Phi)(\Phi 0 \wedge (\forall x)(Nx \supset (\Phi x \supset \Phi(Sn))) \supset (\forall y)(Ny \supset \Phi y))$$

Since Frege, Russell, and Whitehead were interested in reducing arithmetic to logic, they needed a way to translate each of the Peano axioms into logic. So second-order logic seemed like a good way to do this.

8 Other Formal Languages and Logics

It is common to find some other logics and artificial languages in philosophy papers as well. Some of these are further extensions to PL and FOL. For example, modal logics, which are logics used to study different kinds of necessity and possibility, add the symbols $\Box$ and $\Diamond$ to the inventory of symbols. They can be added to the fronts of sentences in any place that a negation can. For example, the sentence $\Box(\forall x)x = x$ could be translated as 'Necessarily, everything is identical to itself'.

There are also some other logics that use PL and FOL as their languages, but which develop different proof systems and semantics in order to validate different inferences. For example, *paraconsistent* logics allow contradictions without grinding to a halt, and *intuitionistic* logics don't validate the law of excluded middle ($P \vee \neg P$). These logics and many others have been motivated by a mixture of philosophical considerations, real-world applications, and mathematical curiosity.