# 4 Second-order logic, quite briefly

Classical first-order logic contrasts along one dimension with various non-classical logics, and along another dimension with second-order and higher-order logics. We'll leave the exploration of non-classical logics to later chapters, starting with Chapter 8. We will, however, say a little about second-order logic straight away, in this chapter. Why?

Theories expressed in first-order languages with a first-order logic turn out to have their limitations – that's a theme that will recur when we look at model theory (Chapter 5), theories of arithmetic (Chapter 6), and set theory (Chapter 7). You will occasionally find explicit contrasts being drawn with richer theories expressed in second-order languages with a second-order logic. So, though it's a judgement call, I think it is worth knowing something early on about second-order logic, in order to understand the contrasts being drawn. At this stage, you need little more than what is sketched in the overview here. You can then either read up a bit further now, or return to get a proper grip on second-order logic later, as and when needed. But first, . . .

## 4.1 A preliminary note on many-sorted logic

(a) As you will now have seen from the core readings, FOL is standardly presented as having a single 'sort' of quantifier, in the sense that all the quantifiers in a given language run over one and the same domain of objects. But this is artificial, and certainly doesn't conform to everyday mathematical practice.

To take an example which will be very familiar to mathematicians, consider the usual practice of using one style of variable for scalars and another for vectors, as in the rule for scalar multiplication:

$$a(\mathbf{v}_1 + \mathbf{v}_2) = a\mathbf{v}_1 + a\mathbf{v}_2. \tag{1}$$

If we want to make the generality here explicit, we could very naturally write

$$\forall a \forall \mathbf{v}_1 \forall \mathbf{v}_2 (\mathbf{v}_1 + \mathbf{v}_2) = a\mathbf{v}_1 + a\mathbf{v}_2, \tag{2}$$

with the first quantifier running just over scalars, and the other two quantifiers running just over vectors. Or we could explicitly declare the domain a quantified variable is running over by using a notation like $(\forall a \colon S)$ to assign $a$ to scalars: mathematicians often do this informally.

It might seem odd, then, to insist that, if we want to formalize our theory of vector spaces, we should follow FOL practice and use only *one* sort of variable and so need to render the rule for scalar multiplication along the lines of

$$\forall x \forall y \forall z((Sx \land Vy \land Vz) \to x(y+z) = xy + xz), \tag{3}$$

i.e. 'Take any three things in our [inclusive] domain, if the first is a scalar, the second is a vector, and the third is a vector, then ... '.

(b) In sum, the theory of vector spaces is naturally regimented using a *two-sorted* logic, with two sorts of variables running over two different domains.

Generalizing, why not allow a *many-sorted* logic – allowing multiple independent domains of objects, with different sorts of variables restricted to running over the different domains? And it isn't hard to set up such a revised version of FOL (it *is* first-order, as the quantifiers are still of the familiar general type, running over objects in the relevant domains). The syntax and semantics of a many-sorted language can be defined quite easily. Syntactically, we will just need to keep a tally of the sorts assigned to the various names and variables, and we will also need rules about which sorts of terms can go into which slots in predicates and in function-expressions. Semantically, we assign a domain for each sort of variable, and then proceed pretty much as in the one-sorted case. Assuming that each domain is non-empty (as in standard FOL) the inference rules for a deductive system will then look entirely familiar. And the resulting logic will have the same nice technical properties as standard one-sorted FOL; crucially, you can prove soundness and completeness and compactness theorems in just the same ways.

(c) As so often in the formalization game, we are now faced with a cost/benefit trade-off. We can get the benefit of somewhat more natural regimentations of mathematical practice, at the cost of having to use a slightly more complex many-sorted logic. Or we can pay the price of having to use less natural regimentations – we need to translate propositions like (2) by using restricted quantifications like (3) – but get the benefit of a slightly-simpler-in-practice logic.[1]

So you pays your money and you takes your choice. For most purposes, logicians prefer the second option, sticking to standard FOL. That's because at the end of the day they care rather less about elegance when regimenting this or that theory than about sticking to a simple-but-powerful logical system.

## 4.2 Second-order logic: a brief overview

(a) Now we turn from 'sorts' to 'orders'. It will help to fix ideas if we begin with an easy arithmetical example; so consider the informal principle of induction:

[1] We also get some added flexibility on the second option. The use of a sorted quantifier $\forall a Fa$ with the usual logic presupposes that there is at least one thing in the relevant domain for the variable $a$. But a corresponding restricted quantification $\forall x(Ax \to Fx)$, where the variable $x$ quantifies over some wider domain, while $A$ picks out the relevant sort which $a$ was supposed to run over, leaves open the possibility that there is nothing of that sort.

(1) Take *any* numerical property $X$; if (i) zero has $X$ and (ii) any number which has $X$ passes it on to its successor, then (iii) *all* numbers must share property $X$.

This holds, of course, because every natural number is either zero or is an eventual successor of zero (i.e. is either 0 or $0'$ or $0''$ or $0'''$ or $\ldots$, where the prime '$'$' is a sign for the function that maps a number to its successor). There are no stray numbers outside that sequence, so a property that percolates down the sequence eventually applies to any number at all.

There is no problem about expressing some particular *instances* of the induction principle in a first-order language. For example, suppose $\mathsf{P}$ is a formal one-place predicate expressing an arithmetical property: then we can express the induction principle for this property by writing

(2) $$(\mathsf{P0} \land \forall \mathsf{x}(\mathsf{Px} \to \mathsf{Px}')) \to \forall \mathsf{x}\, \mathsf{Px}$$

where the small-'x' quantifier runs over the natural numbers and again the prime expresses the successor function. But how can we state the *general* principle of induction in a formal language, the principle that applies to *any* numerical property? The natural candidate is something like this:

(3) $$\forall \mathsf{X}((\mathsf{X0} \land \forall \mathsf{x}(\mathsf{Xx} \to \mathsf{Xx}')) \to \forall \mathsf{x}\, \mathsf{Xx}).$$

Here the big-'X' quantifier is a new type of quantifier, which unlike the small-'x' quantifier, quantifies 'into predicate position'. In other words, it quantifies into the position occupied in (2) by the predicate 'P', and the expressed generalization is intended to run over all *properties* of numbers, so that (3) indeed formally renders (1). But this kind of quantification – *second-order* quantification – is not available in standard first-order languages of the kind that you now know and love.

If we do want to stick with a theory framed in a first-order arithmetical language $L$ which just quantifies over numbers, the best we can do to render the induction principle is to use a template or schema and say something like

(4) For any arithmetical $L$-predicate $\varphi(\ )$, simple or complex, the corresponding sentence $(\varphi(\mathsf{0}) \land \forall \mathsf{x}(\varphi(\mathsf{x}) \to \varphi(\mathsf{x}')) \to \forall \mathsf{x}\, \varphi(\mathsf{x})$ is an axiom.

However (4) is much weaker than the informal (1) or the equivalent formal version (3) on its intended interpretation. For (1/3) tells us that induction holds for *any property at all*; while, in effect, (4) only tells us that induction holds for *those properties that can be expressed by some $L$-predicate $\varphi(\ )$*.

Something like (3), then, is the natural way of formalizing the full principle of induction. So why not allow second-order quantification, quantification into predicate position?

(b) Again, it isn't difficult to extend the syntax and semantics of first-order languages to allow for second-order quantification.

The required added *syntax* is unproblematic. Recall how we can take a formula $\varphi(\mathsf{n})$ containing some occurrence(s) of the name 'n', swap out the name on each occurrence for a particular (small) variable, and then form a first-order quantified wff like $\forall\mathsf{x}\varphi(\mathsf{x})$. We just need now to add the rule that we can take a formula $\varphi(\mathsf{P})$ containing some occurrence(s) of the unary predicate '$\mathsf{P}$', swap out the predicate for some (big) variable and then form a second-order quantified wff like $\forall\mathsf{X}\varphi(\mathsf{X})$. Fine print apart, that's straightforward.

The standard *semantics* is equally straightforward. Again we model the story about the second-order quantifiers on the account of first-order quantifiers. So first fix a domain of quantification. Recall that $\forall\mathsf{x}\varphi(\mathsf{x})$ is true on a given interpretation of its language just when $\varphi(\mathsf{n})$ remains true, however we vary the object in the domain which is assigned to the name '$\mathsf{n}$' as its interpretation. Similarly then, $\forall\mathsf{X}\varphi(\mathsf{X})$ is true on an interpretation just when $\varphi(\mathsf{P})$ remains true, however we vary the subset of the domain which is assigned to the predicate '$\mathsf{P}$' as *its* interpretation (i.e. however we vary '$\mathsf{P}$'s extension). Again, there's fine print; but you get the general idea.

We'll want to expand the syntactic and semantic stories further to allow second-order quantification over binary and other relations and over functions too; but these expansions raise no extra issues.

We can then define the relation of semantic consequence for formulas in our extended languages including second-order quantifiers in the now familiar way: some formulas $\Gamma$ semantically entail $\varphi$ just in case every interpretation over a structure that makes all of $\Gamma$ true makes $\varphi$ true.

(c) So, in bald summary, the situation is this. There are quite a few familiar mathematical claims which, like the arithmetical induction principle, are naturally regimented using quantifications over properties (and/or relations and/or functions). And there is no problem about augmenting the syntax and semantics of our formal languages to allow such second-order quantifications, and we can carry over the definition of semantic entailment to cover sentences in the resulting second-order languages.

Moreover, theories framed in second-order languages turn out to have nice properties which are lacked by their first-order counterparts. For example, a theory of arithmetic with the full second-order induction principle (3) will be 'categorical', in the sense of having just one kind of structure as a model (a model built from a zero, its eventual successors, and nothing else). On the other hand, as you we will see, a first-order theory of arithmetic which has to rely on a limited induction principle like (4) will have models of quite different kinds (as well as the intended model with just a zero and its eventual successors, there will be an infinite number of different 'non-standard' models which have unwanted junk in their domains).

The obvious question which arises, then, is *why have we followed the standard modern practice of privileging FOL*? Why not adopt a second-order logic from the outset as our preferred framework for regimenting mathematical arguments? – after all, as noted in §3.5, early formal logics like Frege's allowed more than first-order quantifiers.

(d) The short answer is: *because there can be no sound and complete formal deductive system for second-order logic.*

There can be be sound but partial deductive systems $S_2$ for a language including second-order quantifiers. So we can have the one-way conditional that, whenever there is an $S_2$-proof from premisses in $\Gamma$ to the conclusion $\varphi$, then $\Gamma$ semantically entails $\varphi$. But the converse fails. We can't have a respectable formal system $S_2$ (where it is decidable what's a proof, etc.) such that, whenever $\Gamma$ semantically entails $\varphi$, there is an $S$-proof from premisses in $\Gamma$ to the conclusion $\varphi$. *Once second-order sentences (with their standard interpretation) are in play, we can't fully capture the relation of semantic entailment in a formal deductive system.*

(e) Let's pause to contrast the case of a two-sorted first-order language of the kind we met in the previous section. In *that* case, the two sorts of quantifier get interpreted quite independently – fixing the domain of one doesn't fix the domain of the other. And it is because each sort of quantifier, as it were, stands alone, a familiar kind of first-order logic continues to each seperately.

But in second-order logic it is entirely different. For note that on the standard semantic story, it is now the *same* domain which fixes the intepretation of both kinds of quantifier – i.e. one and the same domain both provides the objects for the first-order quantifiers to range over, and also provides the sets of objects (i.e. *all* the subsets of the original domain) for the second-order quantifiers to range over. The interpretations of the two kinds of quantifier are tightly connected, and this makes all the difference; it is this which blocks the possibility of a complete deductive system for second-order logic.

(Technical note: If we drop the requirement of standard or 'full' semantics that the second-order big-'X' quantifiers run over *all* the subsets of the domain of the corresponding first-order small-'x' quantifiers, we will arrive what's called 'Henkin semantics' or 'general semantics'. And on this semantics we can regain a completeness theorem, but we lose the nice features that second-order theories have on their natural 'standard' semantics.)

(f) It's not obvious that we *can't* have a complete deductive system for second-order logic with the standard semantics, any more than it is obvious that we *can* have a complete deductive system for first-order logic![2] And it isn't obvious either what the significance this technical result might be. In fact, the whole question of the status of second-order logic leads to a tangled debate.

Let's briefly touch on one thread of that debate. On the usual story, when we give the semantics of FOL, we interpret one-place predicates by assigning them *sets* as extensions. And when we now add second-order quantifiers, we are adding quantifiers which are correspondingly interpreted as ranging over all these possible extensions. So, you might well ask, why not frankly rewrite (3),

[2]There is a rather simple argument that compactness fails in the second-order case, and hence – recycling the ideas of §3.1, fn.2 – second-order logic can't have a *strongly* complete proof system. But it takes much more work to show that we can't even have a *weakly* complete proof system: the usual argument relies on Gödel's incompleteness theorem.

for example, in the form

$$\forall X((0 \in X \land \forall x(x \in X \to x' \in X) \to \forall x\, x \in X), \qquad (5)$$

making it explicit that the big-'X' variable is running over sets? Well, we *can* do that. Though if (5) is to replicate the content of (3) on its standard semantics, it is crucial that the big-'X' variable has to run over *all* the subsets of the domain of the small-'x' variable.

And now some would say that, because (3) can be rewritten as (5), this just goes to show that in using second-order quantifiers we are straying into the realm of set theory. Others would push the connection in the other direction. They would start by arguing that the invocation of sets in the explanation of second-order semantics, while conventional, is actually dispensable (in the spirit of §2.4; and see the papers by Boolos mentioned in the next section). So this means that (5) in fact dresses up the induction principle (3) – which is not in essence set-theoretic – in misleadingly fancy clothing.

So we are left with a troublesome question: is second-order logic really just some "set theory in sheep's clothing" (as the philosopher W.V.O. Quine famously quipped)? We can't pursue this further here (though I give some pointers in the next section for philosophers who want to tackle the issue). Fortunately, for the purposes of getting to grips with the logical material of the next few chapters, you just need to grasp a few basic technical facts about second-order logic: in particular, to stress the point again, while second-order theories can have initially attractive feature, they cannot be pinned down in nicely behaved formal deductive systems.

## 4.3 Recommendations on many-sorted and second-order logic

First, for something on the formal details of many-sorted first-order languages and their logic:

What little you need for present purposes is covered in four clear pages by

1. Herbert Enderton, *A Mathematical Introduction to Logic* (Academic Press 1972, 2002), §4.3.

There is, however, a bit more that can be fussed over here, and some might be interested looking at Hans Halvorson's *The Logic in Philosophy of Science* (CUP, 2019), §§5.1–5.3

Turning now to second-order logic:

For a brief review, saying only a little more than my overview remarks, see

2. Richard Zach and others, *Sets, Logic, Computation*** (Open Logic) §11.3, excerpted at tinyurl.com/openlogicSOL.

You could then look e.g. at the rest of Chapter 4 of the Enderton (1). Or, rather more usefully at this stage, read

3. Stewart Shapiro, 'Higher-order Logic', in S. Shapiro, ed., *The Oxford Handbook of the Philosophy of Mathematics and Logic* (OUP, 2005).

You can skip §3.3; but §3.4 touches on Boolos's ideas and is relevant to the question of how far second-order logic presupposes set theory. Shapiro's §5, 'Logical choice', is an interesting discussion of what's at stake in adopting a second-order logic. (Don't worry if some points will only become really clear once you've done some model theory and some formal arithmetic.)

To nail down some of the technical basics you can then very usefully supplement the explanations in Shapiro with the admirably clear

4. Tim Button and Sean Walsh, *Philosophy and Model Theory** (OUP, 2018), Chapter 1.

This chapter reviews, in a particularly helpful way, various ways of developing the semantics of *first-order* logical languages; and then it compares the first-order case with the *second-order* options, both 'full' semantics and 'Henkin' semantics.

If these initial readings leave you still wanting to fill out the technical story about second-order logic a little further, you will then want to dive into the self-recommending

(4) Stewart Shapiro, *Foundations without Foundationalism: A Case for Second-Order Logic*, Oxford Logic Guides 17 (Clarendon Press, 1991), Chs. 3–5 (with Ch. 6 for enthusiasts).

And philosophers who have Shapiro's wonderfully illuminating book in their hands, will also be intrigued by the initial philosophical/methodological discussion in his first two chapters here. This whole book is a modern classic, and is remarkably accessible.

Shapiro – in both his *Handbook* essay and in his earlier book – mentions Boolos's arguments against treating second-order logic as essentially set-theoretical. Just because he is so very readable, let me highlight the thought-provoking

(5) George Boolos, 'On Second Order Logic' and 'To Be is to Be a Value of a Variable (or to Be Some Values of Some Variables)', both reprinted in his wonderful collection of essays *Logic, Logic, and Logic* (Harvard UP, 1998).

You can then follow up some of the critical discussions of Boolos mentioned by Shapiro.

We return to second- and higher-order logics in Part III.