2 First-Order Predicate Logic

In this chapter, we'll review the syntax and semantics of first-order predicate logic. In contrast to propositional logic, most (some would say: all) mathematical reasoning can be formalized in first-order logic.

2.1 Syntax

I'll begin with a basic version of first-order logic, without function symbols and identity; these will be added in section 2.4. For now, the *primitive symbols* of a first-order language $\mathfrak{L}_1$ therefore fall into the following categories (whose members must not be part of one another):

- a countably infinite set of *(individual) variables*,
- a countably infinite set of *(individual) constants*,
- for each natural number n, a set of *n-ary predicate symbols*,
- the *connectives* '$\neg$' and '$\rightarrow$',
- the *universal quantifier symbol* '$\forall$',
- the parentheses '(' and ')'.

The individual constants and variables constitute the *singular terms* of $\mathfrak{L}_1$. Intuitively, their function is to pick out an object, which might be a person, a number, a set, or anything else. Predicate symbols are used to attribute properties or relations to these objects. For example, we might have individual constants 'a' and 'b' for Athens and Berlin and a binary predicate 'R' for *being west of*. 'Rab' would then state that Athens is west of Berlin, and 'Rba' that Berlin is west of Athens. The predicate symbol always comes first.

From *atomic sentences* like 'Rab' or 'Fa', we can form complex sentences in the familiar way with the help of '$\neg$', '$\rightarrow$', and the parentheses: $\neg Rab$, $(Rab \rightarrow Fa)$, $\neg(Rab \rightarrow Fa)$, etc.

The real power and complexity of first-order logic comes from its quantificational apparatus. The quantifier symbol '$\forall$' allows making general claims about all objects –

where by 'all' I mean all objects in the intended domain of discourse. In a formal theory of arithmetic, for example, the intended domain of discourse would consist of the natural numbers 0, 1, 2, 3, etc. It would not include Athens. In this context, '$\forall xFx$' would state that every natural number has the property expressed by 'F'.

Some practice is required to become familiar with the use of '$\forall$', as it has no direct analog in natural language. The closest translation of '$\forall xFx$' in English is something like

> Everything is such that it is F.

This can obviously be simplified to 'Everything is F'; but in that sentence, 'everything' combines directly with a predicate ('is F'), whereas '$\forall x$' combines with an expression of sentential type, 'Fx'. The variable 'x' works much like the pronoun 'it' in English. Overt variables are sometimes used in English when quantifiers are nested:

> For every number x there is a number y greater than x such that every number greater than y is greater than x.

This can be easily expressed in first-order logic:

$$\forall x \exists y (Gyx \wedge \forall z (Gzy \rightarrow Gzx)).$$

Definition 2.1

A *formula* of a basic first-order language $\mathfrak{L}_1$ is a finite string built up according to the following formation rules:

(i) If P is an n-ary predicate symbol of $\mathfrak{L}_1$ and $t_1, \ldots, t_n$ are singular terms of $\mathfrak{L}_1$ then $Pt_1 \ldots t_n$ is a formula.
(ii) If A is an $\mathfrak{L}_1$-formula, then so is $\neg A$.
(iii) If A and B are $\mathfrak{L}_1$-formulas, then so is $(A \rightarrow B)$.
(iv) If x is a variable and A is a formula of $\mathfrak{L}_1$ then $\forall xA$ is a formula.

Here, 'P', 't_1', 't_n', 'A', 'B', 'x' are metalinguistic variables standing for expressions in $\mathfrak{L}_1$. I haven't specified what the predicate symbols, individual constants, and variables of the object language look like.

As in the case of propositional logic, we introduce some shortcuts in the metalanguage, writing

- $(A \wedge B)$ for $\neg(A \to \neg B)$;
- $(A \vee B)$ for $(\neg A \to B)$;
- $(A \leftrightarrow B)$ for $\neg((A \to B) \to \neg(B \to A))$.
- $\top$ for $A \to A$;
- $\bot$ for $\neg(A \to A)$;
- $\exists x A$ for $\neg \forall x \neg A$.

The last of these is new. We'll omit parentheses and quotation marks when no ambiguity threatens.

Definition 2.1 allows for formulas like these:

$$Rax$$
$$Fa \to Gx$$

If, as before, we interpret 'a' as denoting Athens and 'R' as being west of, 'Rax' could be read as 'Athens is west of x'. But variables, unlike constants, don't pick out a definite object. Their only function is to construct quantified statements. '$\forall x Rax$' would say that Athens is west of everything, but 'Rax' doesn't really say anything. It is neither true nor false.

Formulas like 'Rax' and '$Fa \to Gx$' that contain a variable without a matching quantifier are called "open". Formulas without such variables are "closed". Only closed formulas make a genuine claim about the intended domain of discourse.

Let's make this distinction more precise. A *quantifier* consists of the symbol '$\forall$' followed by a variable. That variable is said to be *bound* by the quantifier. Next, define a *subformula* of a formula as any part of the formula that is itself a formula. For example, Fx is a subformula of $\forall x Fx$. The *scope* of an occurrence of a quantifier $\forall x$ in a formula is the shortest subformula that contains the occurrence. So the scope of $\forall x$ in $Fa \vee \forall x(Fx \to Gy)$ is $\forall x(Fx \to Gy)$. An occurrence of a variable in a formula is *bound* if it lies in the scope of an occurrence of a quantifier that binds it. An occurrence of a variable that isn't bound is *free*. A formula in which some variable occurs free is *open*. A formula that isn't open is *closed*. A *sentence* is a closed formula.

Exercise 2.1 Why do I say that "occurrences" of a variable in a formula are free or bound? Why not simply say that a variable is free or bound in a formula?

Exercise 2.2 Assume that 'F' is a 1-ary (= *monadic*) predicate, 'a' is a constant, and 'y' a variable. Which of the following are formulas? Which are sentences? Mark the scope of each quantifier.
(a) $Fa \to \forall xFx$ (b) $\forall xFx \to Fx$ (c) $\forall xFa$ (d) $\forall x(Fa \to \forall x\neg(Fx \to Fa))$

2.2 The first-order predicate calculus

Frege's *Begriffsschrift* from 1879 contains a complete proof system for first-order logic. However, the formal language of the *Begriffsschrift* is not a first-order language, as it allows quantifying into predicate position. In Frege's language, one can say not only things like '$\forall x\, Fx$', but also '$\forall X\, Xa$', which (roughly) means that a has every property. Quantifiers that bind predicate-type expressions are called *second-order*, and the resulting logic is called *second-order logic*. We'll take a closer look at second-order logics in chapter **??**.

A complete calculus for pure first-order logic was first presented by David Hilbert and Wilhelm Ackermann in 1928. I give a slightly simplified version of their calculus, which I'll call *the first-order predicate calculus*.

Like the propositional calculus from the previous chapter, the first-order calculus consists of some axioms and inference rules. In fact, we'll take over all the axioms and rules of the propositional calculus. All instances of A1-A3 are axioms, and Modus Ponens (MP) is a rule of our new calculus. To these, we add some principles for dealing with quantifiers. Let's think about what we need.

A common inference pattern in first-order logic is "universal instantiation": having shown that *every* object has some property, we infer that a particular object c has that property. To state this precisely, we need some notation for substitution. If A is a formula, x a variable, and c an individual constant, I write '$A(x/c)$' for the formula obtained from A by replacing all free occurrences of x in A with c. For example, '$Fx(x/a)$' denotes the formula 'Fa', but '$\forall xFx(x/a)$' denotes '$\forall xFx$' rather than the nonsensical '$\forall aFa$'. In informal contexts, I'll often write '$A(x)$' to indicate that A is a formula in which the variable x occurs freely; '$A(t)$' is then shorthand for '$A(x/t)$'.

Exercise 2.3 Let A be $\forall x(Fx \to Gy) \to \forall yFy$. What is $A(y/b)$?

We can now formulate a rule of universal instantiation: if A is a formula, x a variable and c a constant, one may infer $A(x/c)$ from $\forall xA$. We won't actually add this as a new

rule, however, because we can just as well add a corresponding axiom schema:

A4 $\forall xA \to A(x/c)$

Given A4, we can use MP to reason from $\forall xA$ to $A(x/c)$.

We introduce a genuine rule – called *(Universal) Generalization* – for the inference in the opposite direction, from a particular case to a general claim:

Gen From A one may infer $\forall xA(c/x)$.

Here, '(c/x)' expresses the inverse of '(x/c)': $A(c/x)$ is the formula obtained from A by replacing all occurrences of c in A with x, except for any occurrences that would let x become bound. (Just as $A(x/c)$ only replaces free occurrences of x, we want $A(c/x)$ to only create free occurrences of x.)

The Gen rule requires explanation. Do we really want to infer $\forall xFx$ from Fa? The inference clearly isn't valid: it's easy to imagine cases where a particular object a is F, but other objects are not F. But remember that each line in a strictly Hilbert-style axiomatic proof is either an axiom or follows from an axiom by an inference rule. None of our axioms or rules will allow making specific claims about any particular object that one couldn't equally make about all other objects. Fa won't be provable. The only provable sentences involving individual constants will be logical truths like $Fa \to Fa$. And here, the inference to $\forall x(Fx \to Fx)$ is safe.

To get a complete calculus, we need one more axiom schema:

A5 $\forall x(A \to B) \to (A \to \forall xB)$, if x is not free in A

To see the point of this, suppose that in the course of a proof we have established the following claims, for some A and $B(x)$, where x isn't free in A:

$\forall x(A \to B(x))$
A

If we had the rule of universal instantiation, we could deduce $A \to B(c)$ from the first line, then use MP to infer $B(c)$ and finally infer $\forall xB(x)$ by Gen. This reasoning can't be replicated after we've replace universal instantiation by the axiom schema A4. So we add A5, which allows inferring $\forall xB(x)$ by two applications of Modus Ponens.

Here's a summary of our axioms and rules:

A1	$A \to (B \to A)$
A2	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
A3	$(\neg A \to \neg B) \to (B \to A)$
A4	$\forall x A \to A(x/c)$
A5	$\forall x(A \to B) \to (A \to \forall x B)$, if x is not free in A
MP	From A and $A \to B$ one may infer B.
Gen	From A one may infer $\forall x A(c/x)$.

Definition 2.2: Proof

A *proof* of a sentence A in the first-order calculus is a finite sequence of sentences $A_1, A_2, \ldots A_n$ with $A_n = A$, such that each A_i is either an instance of A1-A5 or follows from earlier sentences in the sequence by MP or Gen.

I'll use '$\vdash A$' to express that A is provable in the first-order calculus.

As in the case of propositional logic, it is convenient to generalize our Hilbert-style proof system to allow for deductions from premises. In this case, we have to restrict the use of Gen: we don't want to infer $\forall x Fx$ from Fa.

Definition 2.3

If A is a first-order sentence and Γ a set of first-order sentences, a *deduction* of A from Γ in the first-order calculus is a finite sequence of sentences $A_1, A_2, \ldots A_n$, with $A_n = A$, such that each A_i is either an instance of A1–A5, an element of Γ, or follows from previous sentences by MP or Gen, but without applying Gen to an individual constant c that occurs in one of the sentences in the sequence that are elements of Γ.

We write '$\Gamma \vdash A$' to express that there is a deduction of A from Γ. Let's investigate what this relation looks like.

The structural principles Id, Mon, and Cut from the previous chapter hold for every axiomatic calculus. So we have

Id	$A \vdash A$
Mon	If $\Gamma \vdash A$ then $\Gamma, B \vdash A$
Cut	If $\Gamma \vdash A$ and $\Delta, A \vdash B$ then $\Gamma, \Delta \vdash B$

(As in the previous chapter, we generally omit set brackets on the left-hand side of '$\vdash$', and write a comma to indicate unions: '$\Gamma, B \vdash A$' is shorthand for '$\Gamma \cup \{B\} \vdash A$'.)

The Deduction Theorem also still holds:

Theorem 2.1: The Deduction Theorem (DT)

If $\Gamma, A \vdash B$ then $\Gamma \vdash A \to B$.

Proof. Let $B_1, B_2, \ldots, B_n$ be a deduction of B from $\Gamma \cup \{A\}$. We prove by strong induction on k that $\Gamma \vdash A \to B_k$ for all $k = 1, 2, \ldots, n$. That is, we show that *if* $\Gamma \vdash A \to B_i$ for all $i < k$, *then* $\Gamma \vdash A \to B_k$.

We need to distinguish four cases, corresponding to the ways in which B_k can appear in the deduction: as an axiom, as an element of $\Gamma \cup \{A\}$, from an application of MP, or from an application of Gen. The proof for the first three cases is exactly as in the proof of the Deduction Theorem for the propositional calculus. It remains to check the case of Gen.

Assume B_k follows from B_i by an application of Gen. So B_k is of the form $\forall x B_i(c/x)$, and c doesn't occur in Γ or A. By induction hypothesis, there is a deduction of $A \to B_i$ from Γ. As c doesn't occur in Γ, we can apply Gen, getting a deduction of $\forall x(A \to B_i)(c/x)$. Since c doesn't occur in A, this formula can also be written as $\forall x(A \to B_i(c/x))$. A5 gives us $\forall x(A \to B_i(c/x)) \to (A \to \forall x B_i(c/x))$. By MP, we therefore get a deduction of $A \to \forall x B_i(c/x)$ from Γ. □

You may remember that we don't need to invoke A1 and A2 any more once we have DT and MP. Similarly, once we have DT, MP, and Gen, we no longer need A5, as any instance of it can be derived. Here is how.

Assume, as in the statement of A5, that x is not free in A. Let c be a constant that doesn't occur in A or B. Then:

1.	$\forall x(A \to B), A \vdash \forall x(A \to B)$	(Id, Mon)
2.	$\vdash \forall x(A \to B) \to (A \to B(x/c))$	(A4, x not free in A)
3.	$\forall x(A \to B), A \vdash A \to B(x/c)$	(MP, 1, 2)
4.	$\forall x(A \to B), A \vdash A$	(Id)
5.	$\forall x(A \to B), A \vdash B(x/c)$	(MP, 3, 4)
6.	$\forall x(A \to B), A \vdash \forall x B$	(Gen, 5, $B(x/c)(c/x) = B$)
7.	$\forall x(A \to B) \vdash A \to \forall x B$	(DT, 6)
8.	$\vdash \forall x(A \to B) \to (A \to \forall x B)$	(DT, 7)

From A4 and DT, we get the rule of universal instantiation:

Theorem 2.2: Universal Instantiation (UI)

If $\Gamma \vdash \forall x A$ then $\Gamma \vdash A(x/c)$.

Proof. Assume $\Gamma \vdash \forall x A$. By A4, $\vdash \forall x A \to A(x/c)$. So by MP, $\Gamma \vdash A(x/c)$. □

From this (and DT), we can derive any instance of A4. So we won't need to invoke A4 any more.

The derivations of EFQ, DNE, and RAA from the previous chapter all go through as before, and make any appeal to A3 unnecessary.

In fact, we know from the completeness theorem for propositional logic that all truth-functional tautologies are provable from A1–A3 and MP. A *truth-functional tautology* is a sentence that is true on every truth-value assignment to atomic sentences. For example, $Fa \to Fa$ is a truth-functional tautology, and so is $\neg\neg\forall x Fx \to \forall x Fx$.

Theorem 2.3: Tautologies (Taut)

$\vdash A$ whenever A is a truth-functional tautology.

Proof. Consider the propositional language $\mathfrak{L}_0$ whose "sentence letters" are the atomic sentences of the first-order language. By theorem 1.6, (the completeness theorem for propositional logic), every sentence in this language that is true on every truth-value assignment is provable from A1–A3 and MP. □

As in the case of propositional logic, we could use the facts that we have established about $\vdash$: Id, Mon, Cut, DT, UI, Taut, together with MP and Gen, to define a sequent calculus. From this, we could derive the kind of natural deduction or tableau calculus that you have probably learned in your intro logic course. We won't pause to explore these matters.

Exercise 2.4 Show that if $\Gamma \vdash A(x/c)$ then $\Gamma \vdash \exists x A$.

Exercise 2.5 Show that if $\Gamma \vdash \forall x(A \to B)$ then $\Gamma \vdash \forall x A \to \forall x B$.

2.3 Semantics

I've already explained informally how first-order languages are interpreted: individual constants are assumed to pick out objects in the intended domain of discourse; predicate symbols express properties or relations among these objects. We'll now make this more precise.

Our semantics is inspired by the truth-conditional approach to meaning. Plausibly, we can determine the conditions under which an atomic first-order sentences is true by assigning objects to individual constants, and properties or relations to predicate symbols. For example, if we know that 'a' picks out Athens, 'b' Berlin, and 'R' the property of being west of, we can determine that 'Rab' is true in a possible scenario iff Athens is west of Berlin in that scenario.

Now remember that logic abstracts away from the meanings of non-logical expressions. Some premises logically entail a conclusion iff there is no conceivable scenario in which the premises are true and the conclusion false, *under any interpretation of the non-logical vocabulary*. The non-logical parts of a first-order language are its individual constants and predicate symbols. As in the case of propositional logic, we will define a *model* as a structure that contains just enough information about a scenario and an interpretation of the non-logical vocabulary to determine the truth-values of all sentences.

What do you need to know about a scenario S and an interpretation I to figure out whether, say, Rab is true? It would obviously suffice to know (1) which objects are picked out by 'a' and 'b' under I, (2) which relation is expressed by 'R' under I, and (3) whether that relation holds between those two objects in S. But you don't need all that information. It would also suffice to know (1) which objects are picked out by 'a' and 'b' under I, and (2) which pairs of objects in S stand in the relation expressed by 'R' under I. For example, if I told you that 'a' picks out Athens, 'b' Berlin, and 'R' expresses a relation that holds between all and only the following pairs of objects: ⟨Athens, Berlin⟩, ⟨Berlin, Paris⟩, ⟨Paris, Rome⟩, you'd know enough to figure out that 'Rab' is true – although you don't really know what the sentence says or what the scenario is like.

Definition 2.4

A *model* $\mathfrak{M}$ of a first-order language $\mathfrak{L}_1$ consists of

(i) a non-empty set D, called the *domain* or *universe* of $\mathfrak{M}$, and

(ii) an *interpretation function* I that assigns to each individual constant of $\mathfrak{L}_1$ a member of D, and to each n-ary predicate of $\mathfrak{L}_1$ a set of n-tuples from D.

An "n-tuple" is a list of n objects. A 1-tuple is simply an object. So a "set of 1-tuples from D" is a set of members of D a "set of 2-tuples from D" is a set of pairs of members of D, and so on. The set assigned to a predicate is called the *extension* of the predicate.

A model's domain can be arbitrarily large, but it can't be empty. That's because I've stipulated that every first-order language has infinitely many individual constants, and definition 2.4 requires that every such constant be assigned an object in the domain. This wouldn't be possible if the domain were empty. But a single object is enough because we allow that all constants pick out the same object.

It is useful to have an expression for the denotation of a non-logical symbol s in a model $\mathfrak{M}$. I'll use '$[\![s]\!]^{\mathfrak{M}}$'. That is, if $\mathfrak{M}$ is a model with interpretation function I, c is an individual constant and P a predicate, then $[\![c]\!]^{\mathfrak{M}}$ is $I(c)$ and $[\![P]\!]^{\mathfrak{M}}$ is $I(P)$.

Exercise 2.6 We can mimic sentence letters by using zero-ary predicate symbols. For example, if P and Q are zero-ary predicates, then $P \to Q$ is a sentence. We might expect that a model should assign a truth-value to zero-ary predicates. How can we define the truth-values T and F to get this result out of definition 2.4? (Hint: A *0-tuple* is a list of zero objects. There is only one such list: the empty list.)

Next, we define what it takes for a sentence A to be true in a model $\mathfrak{M}$. For atomic sentences, this is easy. 'Rab', for example, is true in $\mathfrak{M}$ iff the pair of objects assigned (by $\mathfrak{M}$) to 'a' and 'b' are in the set assigned to 'R'. For negated sentences and conditionals, we can use the same clauses as in propositional logic. Quantified sentences require a little more thought.

Let $A(x)$ be some formula in which x is free. Under what conditions is $\forall x A(x)$ true in a model $\mathfrak{M}$? As a first shot, one might suggest that $\forall x A(x)$ is true iff $A(c)$ is true for every individual constant c. This is called a *substitutional interpretation* of the quantifier. It assumes that every object in the domain is picked out by some individual constant. Definition 2.4 doesn't guarantee this. It allows for models in which some objects don't have a name, just as most stars and most real numbers don't have a name in English.

What we'll say instead is that $\forall x A(x)$ is true in a model $\mathfrak{M}$ iff $A(c)$ is true in every model that differs from $\mathfrak{M}$ at most in the object it assigns to c, where c is some individual constant that doesn't already occur in $A(x)$. For example, '$\forall x Rax$' is true in $\mathfrak{M}$ iff 'Rab' is true in every model that differs from $\mathfrak{M}$ at most in the object it assigns to 'b'. By varying the interpretation of 'b', we can check whether a stands in R to every object in the domain. For definiteness, we'll say that c is the "alphabetically first" individual constant that doesn't occur in $A(x)$, assuming that the constants come with some alphabetical

order.

This approach to the semantics of quantifiers goes back to Benson Mates. An equally popular alternative, due to Alfred Tarski, states that $\forall x A(x)$ is true in a model $\mathfrak{M}$ iff $A(x)$ is true for every way of assigning an individual to x. This requires defining a truth relation not just between sentences and models, but between sentences, models, and so-called "assignment functions" that assign objects to variables. The two approaches deliver the same results. I use Mates' because it requires slightly less machinery.

Definition 2.5

An $\mathfrak{L}_1$-sentence A is true in a model $\mathfrak{M}$ (for short, $\mathfrak{M} \Vdash A$) iff one of the following conditions holds.

(i) A is an atomic sentence $Pc_1 \ldots c_n$ and $\langle [\![c_1]\!]^{\mathfrak{M}}, \ldots, [\![c_n]\!]^{\mathfrak{M}} \rangle \in [\![P]\!]^{\mathfrak{M}}$.

(ii) A has the form $\neg B$ and $\mathfrak{M} \nVdash B$.

(iii) A has the form $(B \to C)$ and $\mathfrak{M} \nVdash B$ or $\mathfrak{M} \Vdash C$.

(iv) A has the form $\forall x B$ and $\mathfrak{M}' \Vdash B(x/c)$ for every model $\mathfrak{M}'$ that differs from $\mathfrak{M}$ at most in the object assigned to c, where c is the alphabetically first individual constant that does not occur in B.

If A is true in $\mathfrak{M}$, we also say that $\mathfrak{M}$ is a model of A, or that $\mathfrak{M}$ satisfies A, A model satisfies a set of sentences if it satisfies each sentence in the set.

Entailment and validity are defined in terms of satisfaction, as in the previous chapter.

Definition 2.6

A set of sentences Γ *entails* a sentence A (for short, $\Gamma \models A$) iff every model that satisfies Γ also satisfies A.
Sentences A and B are *equivalent* if $A \models B$ and $B \models A$.
A sentence is *valid* (for short, $\models A$) iff it is satisfied by every model.

Exercise 2.7 State the truth conditions for $\exists x A$. That is, complete this sentence: '$\exists x A$ is true in a model $\mathfrak{M}$ iff …'.

Exercise 2.8 Give a countermodel to show that $\forall x(Fx \vee Gx) \not\models \forall x Fx \vee \forall x Gx$.

Exercise 2.9 Show that if x is not free in B then $\forall x(A \to B)$ is equivalent to $\exists xA \to B$.

2.4 Functions and identity

Consider the sentence '$1+2 = 3$'. How could we translate this into a first-order language? We could use a three-place predicate symbol S and write '$S(1, 2, 3)$'. But this isn't ideal. It obscures the structure of the original sentence, which states an identity between $1 + 2$ and 3.

As a first step to remedy this situation, let's introduce a predicate for identity. We'll use '='. So '$=ab$' states that a equals b, in the sense that a and b are the very same object. For legibility, we'll "abbreviate" this as '$a = b$'. We'll also write '$a \neq b$' for '$\neg =ab$'.

Of course, nothing in our earlier definition of first-order languages prevented us from having a predicate '='. The real novelty is that we now classify '=' as a logical expression. This means that its interpretation is held fixed: in every model, '=' is interpreted as the identity relation (on the model's domain). We'll also introduce new rules for reasoning with '='. Before we get to these changes, I want to introduce another addition to our definition of first-order languages that allows forming complex terms like '$1 + 2$'.

Let's think about how such terms work. The expression '$1 + 2$' denotes a number: the number 3. (That's why '$1 + 2 = 3$' is true.) In general, for any numerical terms 'a' and 'b', '$a + b$' denotes a number: the sum of a and b. We can therefore understand the '+' sign as expressing a function that maps a pair of numbers to their sum. So understood, '+' is a *function symbol*. Function symbols are ubiquitous in maths. It's useful to have them in our formal languages as well.

Let me say a few general words on the concept of a function, as it will play an important role throughout these notes. A function, in the mathematical and logical sense, takes one or more objects as input and (typically) returns an object as output. An input to a function is also called an *argument* to the function; the output is called the function's *value* for that argument. The inputs and outputs are usually restricted to a certain class of objects, called the function's *domain* and *codomain*, respectively. For example, the addition function takes two numbers as input and returns a number. The square function takes a single number and returns a number. The inputs and outputs don't need to be numbers. There is an "area" function that takes a country as input and returns its area in (say) square kilometres. And there is a "mother" function that takes a person as input and returns their mother.

If a function has domain X and codomain Y, we say that it is a function *from X to Y*. If all inputs and outputs of a function belong to a set X, we say that it is a function *on X*. So the addition function and the square function are functions on the set of numbers, while the area function is a function from the set of countries to the set of numbers.

Some functions are not defined for all objects in their domain. The division function, for example, takes two numbers as input and returns a number, but it is undefined if the second input is zero. Such functions are called *partial*.

Functions are often associated with a recipe or algorithm for determining the output for a given input. There are well-known algorithms for computing sums or squares. But this isn't part of the modern concept of a function. Any mapping from inputs to outputs is a function, even if there is no recipe for determining the output.

Since functions are just mappings from inputs to outputs, they are fully determined by their values for each input. Consider, for example, the function g on the natural numbers that takes a number x and as input and returns x^2 if Goldbach's conjecture is true and 0 if Goldbach's conjecture is false. Goldbach's conjecture says that every even number greater than 2 is the sum of two primes. It is not known whether the conjecture is true. So we don't know what g returns for inputs other than 0. But we know that g is either identical to the square function or to the constant function that returns 0 for every input. Both of these are trivial to compute. So we know that g is trivial to compute, even though we don't know its value for 1!

Exercise 2.10 Give an example of a function with 3 arguments.

Let's now add function symbols to our first-order languages. These combine with singular terms to form new singular terms. For example, if 'f' is a two-place function symbol and 'a' and 'b' are individual constants, then '$f(a,b)$' is a singular term; it denotes the value of the function f for the arguments a and b. '$f(f(a,b),c)$' is another singular term; it denotes the value of f for the arguments $f(a,b)$ and c. Previously, all singular terms were just individual constants and variables, now they can be arbitrarily complex. So we need a recursive definition.

Definition 2.7

A *(singular) term* of a first-order language $\mathfrak{L}_1^=$ with functions and identity is a finite string conforming to the following formation rules.

- Every variable and every individual constant is a singular term.

- If f is an n-ary function symbol ($n > 0$) and $t_1, \ldots, t_n$ are singular terms then $f(t_1, \ldots, t_n)$ is a singular term.

A singular term is *closed* if it contains no variables. Formulas and sentences are defined exactly as before.

Officially, function symbols are placed in front of their arguments, with parentheses and commas to separate the arguments. By this convention, '$(a \times b) + c$' is written '$+(\times(a, b), c)$'. For the sake of readability, we allow the more familiar infix notation as a metalinguistic "abbreviation".

Exercise 2.11 Write down a first-order sentence expressing Lagrange's Theorem, that every natural number is the sum of four squares. Use a language with individual constants '0', '1', '2', '3', ..., and function symbols '+' and '×' for addition and multiplication.

In the axiomatic calculus, we generalize A4 to allow for closed terms t where we previously had individual constants c:

A4 $\quad \forall x A \rightarrow A(x/t)$.

'(x/t)' is the obvious extension of the substitution notation to closed terms.

We don't need any new axioms or rules for function symbols. But we introduce two new axiom schemas for identity:

A6 $\quad t_1 = t_1$
A7 $\quad t_1 = t_2 \rightarrow (A(x/t_1) \rightarrow A(x/t_2))$

Here, t_1 and t_2 are closed terms and A is a formula in which only x is free, so that all instances of the schemas are closed. A7 is often called *Leibniz' Law*. The idea is that if t_1 and t_2 are the very same object, then anything true of t_1 is also true of t_2.

Exercise 2.12 You may wonder why I didn't write A7 as $t_1 = t_2 \rightarrow (A \rightarrow A(t_1/t_2))$. In response, explain why the following sentence is an instance of A7, but not of the alternative formulation: $a = b \rightarrow (Raa \rightarrow Rab)$.

Exercise 2.13 Show: (a) if $\Gamma \vdash t = s$ then $\Gamma \vdash s = t$, (b) if $\Gamma \vdash t = s$ and $\Gamma \vdash s = r$ then $\Gamma \vdash t = r$.

We also need to adjust our semantics. The definition of a model remains the same as before, except that interpretation functions need to interpret the function symbols. We assume that all function symbols denote total functions on the model's domain.

Definition 2.8

A *model* $\mathfrak{M}$ of a first-order language $\mathfrak{L}_1^=$ with functions and identity consists of

(i) a non-empty set D and

(ii) a function I that assigns

- to each individual constant of $\mathfrak{L}_1^=$ a member of D,
- to each n-ary function symbol of $\mathfrak{L}_1^=$ an n-ary total function on D, and
- to each non-logical n-ary predicate of $\mathfrak{L}_1^=$ a set of n-tuples from D.

As before, we write $[\![s]\!]^{\mathfrak{M}}$ for the denotation of a non-logical symbol s in a model $\mathfrak{M}$. We extend this notation to all singular terms:

Definition 2.9

Let $\mathfrak{M}$ be a model of a first-order language $\mathfrak{L}_1^=$ and I the interpretation function of $\mathfrak{M}$.

(i) For any individual constant c, $[\![c]\!]^{\mathfrak{M}} = I(c)$.

(ii) If f is an n-ary function symbol and $t_1, \ldots, t_n$ are singular terms then $[\![f(t_1, \ldots, t_n)]\!]^{\mathfrak{M}} = I(f)([\![t_1]\!]^{\mathfrak{M}}, \ldots, [\![t_n]\!]^{\mathfrak{M}})$.

With this, satisfaction, entailment, and validity are defined essentially as before. I'll only give the definition of satisfaction:

Definition 2.10

An $\mathfrak{L}_1^=$-sentence A is true in a model $\mathfrak{M}$ (for short, $\mathfrak{M} \Vdash A$) if one of the following conditions holds.

(i) A has the form $t_1 = t_2$ and $[\![t_1]\!]^{\mathfrak{M}} = [\![t_2]\!]^{\mathfrak{M}}$.
(ii) A is any other atomic sentence $Pt_1 \ldots t_n$ and $\langle [\![t_1]\!]^{\mathfrak{M}}, \ldots, [\![t_n]\!]^{\mathfrak{M}} \rangle$ is in $[\![P]\!]^{\mathfrak{M}}$.
(iii) A is of the form $\neg B$ and $\mathfrak{M} \nVdash B$.
(iv) A is of the form $(B \to C)$ and $\mathfrak{M} \nVdash B$ or $\mathfrak{M} \Vdash C$.
(v) A is of the form $\forall x B$ and $\mathfrak{M}' \Vdash B(x/c)$ for every model $\mathfrak{M}'$ that differs from $\mathfrak{M}$ at most in the object assigned to c, where c is the alphabetically first individual constant that does not occur in B.

Exercise 2.14 Explain why $\Vdash a = a$, if a is an individual constant.

Exercise 2.15 Define a model in which '1 + 1 = 2' is true and another in which it is false.

Exercise 2.16 Construct a sentence with '=' as the only predicate symbol that is true in a model $\mathfrak{M}$ iff the domain of $\mathfrak{M}$ has (a) at least two members, (b) at most two members, (c) exactly two members.

Two final comments.

One, it would be nice if we could allow for partial functions and empty domains. The problem is that we would then have to deal with "empty" terms that don't pick out anything. (On an empty domain, every term is empty; if f denotes a partial function, $f(a)$ may be empty.) If t is empty, should we say that $t = t$ be true? What about its negation, $t \neq t$? These questions can be answered in different ways, leading to different versions of *free logic*. A free logic is simply a logic in which terms can be empty.

Two. I've stipulated at the very start of this chapter that a first-order language must have infinitely many individual constants. This, too, is somewhat unsatisfactory. Formalized theories of arithmetic or set theory or Newtonian mechanics, for example, typically don't involve infinitely many non-logical symbols. Indeed, the standard first-order theory of sets has only one non-logical symbol: the membership predicate '$\in$'.

Why, then, did I require infinitely many individual constants? There are two reasons. The first arises in our (Mates-style) semantics of quantifiers: Clause (v) in definition 2.10 interprets $\forall x A$ in terms of $A(x/c)$, where c is a constant that doesn't occur in A. Because quantifiers can be nested without limit, this requires an unending supply of constants: we need two constants for the interpretation of $\forall x \forall y A$, three for $\forall x \forall y \forall z A$, and so on. But

these constants are only used in the internal semantic machinery. Their original denotation in the model is irrelevant: it plays no role in clause (v). A similar point applies to the other reason why we need an unending supply of constants. In our first-order calculus, deriving $\forall x B(x)$ from $\forall x A(x)$ often requires instantiation $\forall x A(x)$ to $A(c)$, deriving $B(c)$, and then applying Gen. Here we also need an unending supply of constants to deal with nested quantifiers. But here, too, the meaning of these constants is irrelevant: they are used to denote "arbitrary" objects.

So we need a large supply of constants to play certain internal roles in our proof system and semantics. Conceptually, these constants resemble variables: they are sometimes called *eigenvariables*. We might have decided to classify them as logical. In later chapters, when I speak of the non-logical expressions of, say, formalized set theory, I will usually ignore the eigenvariables.

2.5 Soundness

We have defined two consequence relations: the proof-theoretic (syntactic) relation $\vdash$, and the model-theoretic (semantic) relation $\models$. How are they related? Can we show that $\Gamma \vdash A$ iff $\Gamma \models A$? We can. The *completeness* direction, from $\Gamma \models A$ to $\Gamma \vdash A$, is hard and will be treated in the next chapter. The *soundness* direction, from $\Gamma \vdash A$ to $\Gamma \models A$, is comparatively easy. As in the propositional case, we only have to verify that all axioms are valid and that the rules preserve validity. There is nothing terribly exciting about this proof, but let's go through it anyway.

We'll need the following lemmas.

Lemma 2.1: Coincidence Lemma

If two models $\mathfrak{M}$ and $\mathfrak{M}'$ have the same domain and agree on the interpretation of all non-logical symbols in an $\mathscr{L}_1^=$-sentence A, then $\mathfrak{M} \Vdash A$ iff $\mathfrak{M}' \Vdash A$.

Proof. The proof is a simple induction on the complexity of A. The base case is guaranteed by clauses (i) and (ii) in definition 2.10. The inductive step for $\neg$ and $\rightarrow$ is trivial. Let's look at the case where A has the form $\forall x B$.

Assume $\mathfrak{M} \nVdash \forall x B$. By clause (v) of definition 2.10, this means that $\mathfrak{M}'' \nVdash B(x/c)$ for some model $\mathfrak{M}''$ that differs from $\mathfrak{M}$ at most in the object assigned to a constant c that does not occur in B. Let $\mathfrak{M}'''$ be like $\mathfrak{M}'$ except that $[\![c]\!]^{\mathfrak{M}'''} = [\![c]\!]^{\mathfrak{M}''}$. Since $\mathfrak{M}$ and $\mathfrak{M}'$ agree on all symbols in B, and c is not in B, $\mathfrak{M}''$ and $\mathfrak{M}'''$ agree on all symbols in

$B(x/c)$. So by induction hypothesis, $\mathfrak{M}''' \not\models B(x/c)$. By clause (v) of definition 2.10, this means that $\mathfrak{M}' \not\models \forall x B$.

We've shown that if $\mathfrak{M} \not\models \forall x B$ then $\mathfrak{M}' \not\models \forall x B$. The converse direction can be shown by an exactly parallel argument. So $\mathfrak{M} \models \forall x B$ iff $\mathfrak{M}' \models \forall x B$. □

Lemma 2.2: Extensionality Lemma

If $[\![t_1]\!]^{\mathfrak{M}} = [\![t_2]\!]^{\mathfrak{M}}$ then $\mathfrak{M} \models A(x/t_1)$ iff $\mathfrak{M} \models A(x/t_2)$.

Proof. The proof is by induction on complexity of A. As before, the base case is guaranteed by clauses (i) and (ii) in definition 2.10, and the inductive step for $\neg$ and $\rightarrow$ is trivial. The case where A has the form $\forall y B$ needs some work.

Assume

$$\mathfrak{M} \not\models \forall y B(x/t_1). \tag{1}$$

We'll show that $\mathfrak{M} \not\models \forall y B(x/t_2)$. By (1) and definition 2.10, we have

$$\mathfrak{M}^c \not\models B(x/t_1)(y/c), \tag{2}$$

where c is the alphabetically first constant that does not occur in $B(x/t_1)$ and $\mathfrak{M}^c$ is a model that differs from $\mathfrak{M}$ at most in the interpretation of c. Let d be a constant distinct from c that does not occur in $B(x/t_1)$ or $B(x/t_2)$. let $\mathfrak{M}^{dc}$ be like $\mathfrak{M}^c$ except that $[\![d]\!]^{\mathfrak{M}^{dc}} = [\![c]\!]^{\mathfrak{M}^c}$. Since $\mathfrak{M}^{dc}$ and $\mathfrak{M}^c$ agree on all symbols in $B(x/t_1)(y/c)$, we have, by the coincidence lemma,

$$\mathfrak{M}^c \models B(x/t_1)(y/c) \text{ iff } \mathfrak{M}^{dc} \models B(x/t_1)(y/c). \tag{3}$$

By induction hypothesis,

$$\mathfrak{M}^{dc} \models B(x/t_1)(y/c) \text{ iff } \mathfrak{M}^{dc} \models B(x/t_1)(y/d). \tag{4}$$

Let $\mathfrak{M}^d$ be like $\mathfrak{M}^{dc}$ except that $[\![c]\!]^{\mathfrak{M}^d} = [\![c]\!]^{\mathfrak{M}}$. Since c does not occur in $B(x/t_1)$ and is distinct from d, $\mathfrak{M}^d$ and $\mathfrak{M}^{dc}$ agree on all symbols in $B(x/t_1)(y/d)$. So by the coincidence lemma,

$$\mathfrak{M}^{dc} \models B(x/t_1)(y/d) \text{ iff } \mathfrak{M}^d \models B(x/t_1)(y/d) \tag{5}$$

Since $\mathfrak{M}^d$ agrees with $\mathfrak{M}$ on the interpretation of t_1 and t_2, $[\![t_1]\!]^{\mathfrak{M}^d} = [\![t_2]\!]^{\mathfrak{M}^d}$. So by

induction hypothesis,

$$\mathfrak{M}^d \Vdash B(x/t_1)(y/d) \text{ iff } \mathfrak{M}^d \Vdash B(x/t_2)(y/d). \tag{6}$$

From (2)–(6), we get

$$\mathfrak{M}^d \nVdash B(x/t_2)(y/d). \tag{7}$$

Now let e be the alphabetically first constant that does not occur in $B(x/t_2)$. Assume first that e is distinct from d. Let $\mathfrak{M}^{ed}$ be like $\mathfrak{M}^d$ except that $[\![e]\!]^{\mathfrak{M}^{ed}} = [\![d]\!]^{\mathfrak{M}^d}$. Since e doesn't occur in $B(x/t_2)(y/d)$, $\mathfrak{M}^{ed}$ and $\mathfrak{M}^d$ agree on all symbols in $B(x/y_2)(y/d)$. So by the coincidence lemma,

$$\mathfrak{M}^d \Vdash B(x/t_2)(y/d) \text{ iff } \mathfrak{M}^{ed} \Vdash B(x/y_2)(y/d). \tag{8}$$

By induction hypothesis,

$$\mathfrak{M}^{ed} \Vdash B(x/t_2)(y/d) \text{ iff } \mathfrak{M}^{ed} \Vdash B(x/t_2)(y/e). \tag{9}$$

Finally, let $\mathfrak{M}^e$ be like $\mathfrak{M}$ except that $[\![e]\!]^{\mathfrak{M}^e} = [\![e]\!]^{\mathfrak{M}^{ed}}$. By the coincidence lemma,

$$\mathfrak{M}^{ed} \Vdash B(x/t_2)(y/e) \text{ iff } \mathfrak{M}^e \Vdash B(x/t_2)(y/e). \tag{10}$$

From (7), (8), (9), and (10), we get

$$\mathfrak{M}^e \nVdash B(x/t_2)(y/e). \tag{11}$$

We assumed that e is distinct from d. If e and d are the same constant, we get (11) directly from (7). From (11) and definition 2.10, we conclude that

$$\mathfrak{M} \nVdash \forall y B(x/t_2). \tag{12}$$

We've shown that if $\mathfrak{M} \nVdash \forall y B(x/t_1)$ then $\mathfrak{M} \nVdash \forall y B(x/t_2)$. Swapping t_1 and t_2 throughout the argument, we can equally show that if $\mathfrak{M} \nVdash \forall y B(x/t_2)$ then $\mathfrak{M} \nVdash \forall y B(x/t_1)$. So $\mathfrak{M} \Vdash \forall y B(x/t_1)$ iff $\mathfrak{M} \Vdash \forall y B(x/t_2)$. □

Theorem 2.4: Soundness of the first-order calculus

If $\Gamma \vdash A$, then $\Gamma \models A$.

Proof. We first show a special case: if $\vdash A$ then $\models A$.

Assume $\vdash A$. So there is a sequence $A_1, \ldots, A_n$ with $A_n = A$ such that each A_k in the sequence is either an axiom or follows from previous sentences by MP or Gen. We show by strong induction on k that $\models A_k$.

Case 1. A_k is an instance of A1–A3. Then $\models A_k$ by exercise 1.14 and the fact that the interpretation of '$\neg$' and '$\rightarrow$' in definition 2.10 is the same as in propositional logic.

Case 2. A_k is an instance of A4: $\forall x\, B \rightarrow B(x/t)$. Let $\mathfrak{M}$ be any model that satisfies $\forall x B$. By definition 2.10, this means that $\mathfrak{M}' \Vdash B(x/c)$ for every model $\mathfrak{M}'$ that differs from $\mathfrak{M}$ at most in the object assigned to c, where c does not occur in B. Let $\mathfrak{M}'$ be a model of this kind with $[\![c]\!]^{\mathfrak{M}'} = [\![t]\!]^{\mathfrak{M}'}$. Since $B(x/t)$ is obtained from $B(x/c)$ by substituting t for c, it follows by the extensionality lemma that $\mathfrak{M}' \Vdash B(x/t)$. Finally, since c does not occur in $B(x/t)$, $\mathfrak{M}'$ and $\mathfrak{M}$ agree on the interpretation of all symbols in $B(x/t)$. So $\mathfrak{M} \Vdash B(x/t)$ by the coincidence lemma. This shows that any model that satisfies $\forall x B$ also satisfies $B(x/t)$. Hence every model satisfies $\forall x B \rightarrow B(x/t)$.

Case 3. A_k is an instance of A5: $\forall x(A \rightarrow B) \rightarrow (A \rightarrow \forall x\, B)$, where x is not free in A. Let $\mathfrak{M}$ be any model that doesn't satisfy $A \rightarrow \forall x B$. By definition 2.10, this means that $\mathfrak{M} \Vdash A$ and $\mathfrak{M}' \nVdash B(x/c)$ for some model $\mathfrak{M}'$ that differs from $\mathfrak{M}$ at most in the object assigned to some constant c that does not occur in B. Let d be the alphabetically first constant that does not occur in either A or B, and let $\mathfrak{M}''$ be like $\mathfrak{M}'$ except that $[\![d]\!]^{\mathfrak{M}''} = [\![c]\!]^{\mathfrak{M}'}$. By the extensionality lemma, $\mathfrak{M}'' \nVdash B(x/d)$. By the coincidence lemma, $\mathfrak{M}'' \Vdash A$. So $\mathfrak{M}'' \nVdash A \rightarrow B(x/d)$. Since x is not free in A, $A \rightarrow B(x/d)$ is $(A \rightarrow B)(x/d)$. So $\mathfrak{M}'' \nVdash (A \rightarrow B)(x/d)$. By definition 2.10, this means that $\mathfrak{M} \nVdash \forall x(A \rightarrow B)$. Contraposing, we've shown that any model that satisfies $\forall x(A \rightarrow B)$ satisfies $A \rightarrow \forall x B$. So every model satisfies $\forall x(A \rightarrow B) \rightarrow (A \rightarrow \forall x B)$.

Case 4. A_k is an instance of A6: $t_1 = t_1$. Then $\models A_k$ by clause (i) of definition 2.10.

Case 5. A_k is an instance of A7: $t_1 = t_2 \rightarrow (A(x/t_1) \rightarrow A(x/t_2))$. Let $\mathfrak{M}$ be any model that satisfies $t_1 = t_2$. Then $[\![t_1]\!]^{\mathfrak{M}} = [\![t_2]\!]^{\mathfrak{M}}$. By the extensionality lemma, $\mathfrak{M} \Vdash A(x/t_1)$ iff $\mathfrak{M} \Vdash A(x/t_2)$. So any model that satisfies $t_1 = t_2$ and $A(x/t_1)$ also satisfies $A(x/t_2)$. So every model satisfies $t_1 = t_2 \rightarrow (A(x/t_1) \rightarrow A(x/t_2))$.

Case 6. A_k is obtained by MP from earlier lines A_i and $A_i \rightarrow A_k$. By induction hypothesis, A_i and $A_i \rightarrow A_k$ are valid. So A_k is valid by clause (iii) of definition 2.10.

Case 7. A_k is obtained by Gen from an earlier line A_i. So A_k has the form $\forall x A_i(c/x)$. Let $\mathfrak{M}$ be any model. By definition 2.10, we need to show that $\mathfrak{M}' \Vdash A_i(c/x)(x/d)$ for every model $\mathfrak{M}'$ that differs from $\mathfrak{M}$ at most in the object assigned to d, where

d is the alphabetically first individual constant that does not occur in $A_i(c/x)$. Take any such $\mathfrak{M}'$ and d. Let $\mathfrak{M}''$ be like $\mathfrak{M}'$ except that $[\![c]\!]^{\mathfrak{M}''} = [\![d]\!]^{\mathfrak{M}'}$. By induction hypothesis, every model satisfies A_i; so $\mathfrak{M}'' \Vdash A_i$. If d is c then $\mathfrak{M}'' = \mathfrak{M}'$. Otherwise c does not occur in $A_i(c/x)(x/d)$. Either way, $\mathfrak{M}''$ and $\mathfrak{M}'$ agree on the interpretation of every symbol in $A_i(c/x)(x/d)$. By the coincidence lemma, it follows that $\mathfrak{M}' \Vdash A_i(c/x)(x/d)$.

This completes the induction. We've shown that if $\vdash A$ then $\models A$. Now assume $\Gamma \vdash A$. That is, there is a deduction A from Γ. This deduction can involve only finitely many sentences $A_1, \ldots, A_n$ from Γ. So we also have $A_1, \ldots, A_n \vdash A$, By the deduction theorem, it follows that $\vdash A_1 \to (\ldots (A_n \to A) \ldots)$. From what we've just shown, we can infer that $\models A_1 \to (\ldots (A_n \to A) \ldots)$. It is easy to see that $\Gamma \models A \to B$ iff $\Gamma, A \models B$. So we have $A_1, \ldots, A_n \models A$ and thereby $\Gamma \models A$. □

Exercise 2.17 Soundness implies consistency: it is impossible to prove $\bot$ in the first-order calculus. Recall that a calculus is Post-complete if any addition of a new axiom schema (that is not already provable in the calculus) makes the calculus inconsistent. Can you outline a proof showing that the first-order calculus is not Post-complete?