Part I

Second-order Logic

This is the beginnings of a part on second-order logic.

Chapter 1

Syntax and Semantics

Basic syntax and semantics for SOL covered so far. As a chapter it's too short. Substitution for second-order variables has to be covered to be able to talk about derivation systems for SOL, and there's some subtle issues there.

1.1 Introduction

sol:syn:int: sec

In first-order logic, we combine the non-logical symbols of a given language, i.e., its constant symbols, function symbols, and predicate symbols, with the logical symbols to express things about first-order structures. This is done using the notion of satisfaction, which relates a structure $\mathfrak{M}$, together with a variable assignment s, and a formula φ: $\mathfrak{M}, s \vDash \varphi$ holds iff what φ expresses when its constant symbols, function symbols, and predicate symbols are interpreted as $\mathfrak{M}$ says, and its free variables are interpreted as s says, is true. The interpretation of the identity predicate $=$ is built into the definition of $\mathfrak{M}, s \vDash \varphi$, as is the interpretation of $\forall$ and $\exists$. The former is always interpreted as the identity relation on the domain $|\mathfrak{M}|$ of the structure, and the quantifiers are always interpreted as ranging over the entire domain. But, crucially, quantification is only allowed over elements of the domain, and so only object variables are allowed to follow a quantifier.

In second-order logic, both the language and the definition of satisfaction are extended to include free and bound function and predicate variables, and quantification over them. These variables are related to function symbols and predicate symbols the same way that object variables are related to constant symbols. They play the same role in the formation of terms and formulas of second-order logic, and quantification over them is handled in a similar way. In the *standard* semantics, the second-order quantifiers range over all possible objects of the right type (n-place functions from $|\mathfrak{M}|$ to $|\mathfrak{M}|$ for function variables, n-place relations for predicate variables). For

instance, while $\forall v_0 (P^1_0(v_0) \vee \neg P^1_0(v_0))$ is a formula in both first- and second-order logic, in the latter we can also consider $\forall V^1_0 \forall v_0 (V^1_0(v_0) \vee \neg V^1_0(v_0))$ and $\exists V^1_0 \forall v_0 (V^1_0(v_0) \vee \neg V^1_0(v_0))$. Since these contain no free varaibles, they are sentences of second-order logic. Here, V^1_0 is a second-order 1-place predicate variable. The allowable interpretations of V^1_0 are the same that we can assign to a 1-place predicate symbol like P^1_0, i.e., subsets of $|\mathfrak{M}|$. Quantification over them then amounts to saying that $\forall v_0 (V^1_0(v_0) \vee \neg V^1_0(v_0))$ holds for all ways of assigning a subset of $|\mathfrak{M}|$ as the value of V^1_0, or for at least one. Since every set either contains or fails to contain a given object, both are true in any structure.

1.2 Terms and Formulas

sol:syn:frm:sec

Like in first-order logic, expressions of second-order logic are built up from a basic vocabulary containing *variables*, *constant symbols*, *predicate symbols* and sometimes *function symbols*. From them, together with logical connectives, quantifiers, and punctuation symbols such as parentheses and commas, *terms* and *formulas* are formed. The difference is that in addition to variables for objects, second-order logic also contains variables for relations and functions, and allows quantification over them. So the logical symbols of second-order logic are those of first-order logic, plus:

1. A denumerable set of second-order relation variables of every arity n: V^n_0, V^n_1, V^n_2, ...
2. A denumerable set of second-order function variables: u^n_0, u^n_1, u^n_2, ...

Just as we use x, y, z as meta-variables for first-order variables v_i, we'll use X, Y, Z, etc., as metavariables for V^n_i and u, v, etc., as meta-variables for u^n_i.

explanation

The non-logical symbols of a second-order language are specified the same way a first-order language is: by listing its constant symbols, function symbols, and predicate symbols

In first-order logic, the identity predicate $=$ is usually included. In first-order logic, the non-logical symbols of a language $\mathcal{L}$ are crucial to allow us to express anything interesting. There are of course sentences that use no non-logical symbols, but with only $=$ it is hard to say anything interesting. In second-order logic, since we have an unlimited supply of relation and function variables, we can say anything we can say in a first-order language even without a special supply of non-logical symbols.

Definition 1.1 (Second-order Terms). The set of *second-order terms* of $\mathcal{L}$, $\mathrm{Trm}^2(\mathcal{L})$, is defined by adding to **??** the clause

1. If u is an n-place function variable and $t_1, \ldots, t_n$ are terms, then $u(t_1, \ldots, t_n)$ is a term.

So, a second-order term looks just like a first-order term, except that where a first-order term contains a function symbol f_i^n, a second-order term may contain a function variable u_i^n in its place. explanation

Definition 1.2 (Second-order formula). The set of *second-order formulas* $\mathrm{Frm}^2(\mathcal{L})$ of the language $\mathcal{L}$ is defined by adding to **??** the clauses

1. If X is an n-place predicate variable and $t_1, \ldots, t_n$ are second-order terms of $\mathcal{L}$, then $X(t_1, \ldots, t_n)$ is an atomic formula.
2. If φ is a formula and u is a function variable, then $\forall u\, \varphi$ is a formula.
3. If φ is a formula and X is a predicate variable, then $\forall X\, \varphi$ is a formula.
4. If φ is a formula and u is a function variable, then $\exists u\, \varphi$ is a formula.
5. If φ is a formula and X is a predicate variable, then $\exists X\, \varphi$ is a formula.

1.3 Satisfaction

sol:syn:sat:sec To define the satisfaction relation $\mathfrak{M}, s \vDash \varphi$ for second-order formulas, we have to extend the definitions to cover second-order variables. The notion of a structure is the same for second-order logic as it is for first-order logic. There is only a diffence for variable assignments s: these now must not just provide values for the first-order variables, but also for the second-order variables. explanation

Definition 1.3 (Variable Assignment). A *variable assignment* s for a structure $\mathfrak{M}$ is a function which maps each

1. object variable v_i to an element of $|\mathfrak{M}|$, i.e., $s(v_i) \in |\mathfrak{M}|$
2. n-place relation variable V_i^n to an n-place relation on $|\mathfrak{M}|$, i.e., $s(V_i^n) \subseteq |\mathfrak{M}|^n$;
3. n-place function variable u_i^n to an n-place function from $|\mathfrak{M}|$ to $|\mathfrak{M}|$, i.e., $s(u_i^n)\colon |\mathfrak{M}|^n \to |\mathfrak{M}|$;

A structure assigns a value to each constant symbol and function symbol, and a second-order variable assigns objects and functions to each object and function variable. Together, they let us assign a value to every term. explanation

Definition 1.4 (Value of a Term). If t is a term of the language $\mathcal{L}$, $\mathfrak{M}$ is a structure for $\mathcal{L}$, and s is a variable assignment for $\mathfrak{M}$, the *value* $\mathrm{Val}_s^{\mathfrak{M}}(t)$ is defined as for first-order terms, plus the following clause:

$t \equiv u(t_1, \ldots, t_n)$:

$$\mathrm{Val}_s^{\mathfrak{M}}(t) = s(u)(\mathrm{Val}_s^{\mathfrak{M}}(t_1), \ldots, \mathrm{Val}_s^{\mathfrak{M}}(t_n)).$$

Definition 1.5 (x-Variant). If s is a variable assignment for a structure $\mathfrak{M}$, then any variable assignment s' for $\mathfrak{M}$ which differs from s at most in what it assigns to x is called an *x-variant* of s. If s' is an x-variant of s we write $s \sim_x s'$. (Similarly for second-order variables X or u.)

Definition 1.6 (Satisfaction). For second-order formulas φ, the definition of satisfaction is like **??** with the addition of:

1. $\varphi \equiv X^n(t_1, \ldots, t_n)$: $\mathfrak{M}, s \vDash \varphi$ iff $\langle \mathrm{Val}^{\mathfrak{M}}_s(t_1), \ldots, \mathrm{Val}^{\mathfrak{M}}_s(t_n)\rangle \in s(X^n)$.
2. $\varphi \equiv \forall X\, \psi$: $\mathfrak{M}, s \vDash \varphi$ iff for every X-variant s' of s, $\mathfrak{M}, s' \vDash \psi$.
3. $\varphi \equiv \exists X\, \psi$: $\mathfrak{M}, s \vDash \varphi$ iff there is an X-variant s' of s so that $\mathfrak{M}, s' \vDash \psi$.
4. $\varphi \equiv \forall u\, \psi$: $\mathfrak{M}, s \vDash \varphi$ iff for every u-variant s' of s, $\mathfrak{M}, s' \vDash \psi$.
5. $\varphi \equiv \exists u\, \psi$: $\mathfrak{M}, s \vDash \varphi$ iff there is an u-variant s' of s so that $\mathfrak{M}, s' \vDash \psi$.

Example 1.7. Consider the formula $\forall z\, (X(z) \leftrightarrow \neg Y(z))$. It contains no second-order quantifiers, but does contain the second-order variables X and Y (here understood to be one-place). The corresponding first-order sentence $\forall z\, (P(z) \leftrightarrow \neg R(z))$ says that whatever falls under the interpretation of P does not fall under the interpretation of R and vice versa. In a structure, the interpretation of a predicate symbol P is given by the interpretation $P^{\mathfrak{M}}$. But for second-order variables like X and Y, the interpretation is provided, not by the structure itself, but by a variable assignment. Since the second-order formula is not a sentence (in includes free variables X and Y), it is only satisfied relative to a structure $\mathfrak{M}$ together with a variable assignment s.

$\mathfrak{M}, s \vDash \forall z\, (Xz \leftrightarrow \neg Yz)$ whenever the elements of $s(X)$ are not elements of $s(Y)$, and vice versa, i.e., iff $s(Y) = |\mathfrak{M}| \setminus s(X)$. So for instance, take $|\mathfrak{M}| = \{1, 2, 3\}$. Since no predicate symbols, function symbols, or constant symbols are involved, the domain of $\mathfrak{M}$ is all that is relevant. Now for $s_1(X) = \{1, 2\}$ and $s_1(Y) = \{3\}$, we have $\mathfrak{M}, s_1 \vDash \forall z\, (X(z) \leftrightarrow \neg Y(z))$.

By contrast, if we have $s_2(X) = \{1, 2\}$ and $s_2(Y) = \{2, 3\}$, $\mathfrak{M}, s_2 \nvDash \forall z\, (X(z) \leftrightarrow \neg Y(z))$. That's because there is a z-variant s_2' of s_2 with $s_2'(z) = 2$ where $\mathfrak{M}, s_2' \vDash X(z)$ (since $2 \in s_2'(X)$) but $\mathfrak{M}, s_2' \nvDash \neg Y(z)$ (since also $s_2'(z) \in s_2'(Y)$).

Example 1.8. $\mathfrak{M}, s \vDash \exists Y\, (\exists y\, Y(y) \wedge \forall z\, (X(z) \leftrightarrow \neg Y(z)))$ if there is an $s' \sim_Y s$ such that $\mathfrak{M}, s' \vDash (\exists y\, Y(y) \wedge \forall z\, (X(z) \leftrightarrow \neg Y(z)))$. And that is the case iff $s'(Y) \neq \emptyset$ (so that $\mathfrak{M}, s' \vDash \exists y\, Y(y)$) and, as in the previous example, $s'(Y) = |\mathfrak{M}| \setminus s'(X)$. In other words, $\mathfrak{M}, s \vDash \exists Y\, (\exists y\, Y(y) \wedge \forall z\, (X(z) \leftrightarrow \neg Y(z)))$ iff $|\mathfrak{M}| \setminus s(X)$ is non-empty, i.e., $s(X) \neq |\mathfrak{M}|$. So, the formula is satisfied, e.g., if $|\mathfrak{M}| = \{1, 2, 3\}$ and $s(X) = \{1, 2\}$, but not if $s(X) = \{1, 2, 3\} = |\mathfrak{M}|$.

Since the formula is not satisfied whenever $s(X) = |\mathfrak{M}|$, the sentence

$$\forall X\, \exists Y\, (\exists y\, Y(y) \wedge \forall z\, (X(z) \leftrightarrow \neg Y(z)))$$

is never satisfied: For any structure $\mathfrak{M}$, the assignment $s(X) = |\mathfrak{M}|$ will make the sentence false. On the other hand, the sentence

$$\exists X\, \exists Y\, (\exists y\, Y(y) \land \forall z\, (X(z) \leftrightarrow \neg Y(z)))$$

is satisfied relative to any assignment s, since we can always find an X-variant s' of s with $s'(X) \neq |\mathfrak{M}|$.

1.4 Semantic Notions

sol:syn:sem:sec

The central logical notions of *validity*, *entailment*, and *satisfiability* are defined the same way for second-order logic as they are for first-order logic, except that the underlying satisfaction relation is now that for second-order formulas. A second-order sentence, of course, is a formula in which all variables, including predicate and function variables, are bound. explanation

Definition 1.9 (Validity). A sentence φ is *valid*, $\vDash \varphi$, iff $\mathfrak{M} \vDash \varphi$ for every structure $\mathfrak{M}$.

Definition 1.10 (Entailment). A set of sentences Γ *entails* a sentence φ, $\Gamma \vDash \varphi$, iff for every structure $\mathfrak{M}$ with $\mathfrak{M} \vDash \Gamma$, $\mathfrak{M} \vDash \varphi$.

Definition 1.11 (Satisfiability). A set of sentences Γ is *satisfiable* if $\mathfrak{M} \vDash \Gamma$ for some structure $\mathfrak{M}$. If Γ is not satisfiable it is called *unsatisfiable*.

1.5 Expressive Power

sol:syn:exp:sec

Quantification over second-order variables is responsible for an immense increase in the expressive power of the language over that of first-order logic. Second-order existential quantification lets us say that functions or relations with certain properties exists. In first-order logic, the only way to do that is to specify a non-logical symbol (i.e., a function symbol or predicate symbol) for this purpose. Second-order universal quantification lets us say that all subsets of, relations on, or functions from the domain to the domain have a property. In first-order logic, we can only say that the subsets, relations, or functions assigned to one of the non-logical symbols of the language have a property. And when we say that subsets, relations, functions exist that have a property, or that all of them have it, we can use second-order quantification in specifying this property as well. This lets us define relations not definable in first-order logic, and express properties of the domain not expressible in first-order logic. explanation

Definition 1.12. If $\mathfrak{M}$ is a structure for a language $\mathcal{L}$, a relation $R \subseteq |\mathfrak{M}|^2$ is *definable* in $\mathcal{L}$ if there is some formula $\varphi_R(x, y)$ with only the variables x and y free, such that $R(a, b)$ holds (i.e., $\langle a, b\rangle \in R$) iff $\mathfrak{M}, s \vDash \varphi_R(x, y)$ for $s(x) = a$ and $s(y) = b$.

Example 1.13. In first-order logic we can define the identity relation $\mathrm{Id}_{|\mathfrak{M}|}$ (i.e., $\{\langle a, a\rangle : a \in |\mathfrak{M}|\}$) by the formula $x = y$. In second-order logic, we can define this relation *without* $=$. For if a and b are the same element of $|\mathfrak{M}|$, then they are elements of the same subsets of $|\mathfrak{M}|$ (since sets are determined by their elements). Conversely, if a and b are different, then they are not elements of the same subsets: e.g., $a \in \{a\}$ but $b \notin \{a\}$ if $a \neq b$. So "being elements of the same subsets of $|\mathfrak{M}|$" is a relation that holds of a and b iff $a = b$. It is a relation that can be expressed in second-order logic, since we can quantify over all subsets of $|\mathfrak{M}|$. Hence, the following formula defines $\mathrm{Id}_{|\mathfrak{M}|}$:

$$\forall X\,(X(x) \leftrightarrow X(y))$$

Problem 1.1. Show that $\forall X\,(X(x) \rightarrow X(y))$ (note: $\rightarrow$ not $\leftrightarrow$!) defines $\mathrm{Id}_{|\mathfrak{M}|}$.

Example 1.14. If R is a two-place predicate symbol, $R^{\mathfrak{M}}$ is a two-place relation on $|\mathfrak{M}|$. Perhaps somewhat confusingly, we'll use R as the predicate symbol for R and for the relation $R^{\mathfrak{M}}$ itself. The *transitive closure* R^* of R is the relation that holds between a and b iff for some $c_1, \ldots, c_k$, $R(a, c_1)$, $R(c_1, c_2)$, $\ldots$, $R(c_k, b)$ holds. This includes the case if $k = 0$, i.e., if $R(a, b)$ holds, so does $R^*(a, b)$. This means that $R \subseteq R^*$. In fact, R^* is the smallest relation that includes R and that is transitive. We can say in second-order logic that X is a transitive relation that includes R:

$$\psi_R(X) \equiv \forall x\,\forall y\,(R(x, y) \rightarrow X(x, y)) \wedge \forall x\,\forall y\,\forall z\,((X(x, y) \wedge X(y, z)) \rightarrow X(x, z)).$$

The first conjunct says that $R \subseteq X$ and the second that X is transitive.

To say that X is the smallest such relation is to say that it is itself included in every relation that includes R and is transitive. So we can define the transitive closure of R by the formula

$$R^*(X) \equiv \psi_R(X) \wedge \forall Y\,(\psi_R(Y) \rightarrow \forall x\,\forall y\,(X(x, y) \rightarrow Y(x, y))).$$

We have $\mathfrak{M}, s \vDash R^*(X)$ iff $s(X) = R^*$. The transitive closure of R cannot be expressed in first-order logic.

1.6 Describing Infinite and Enumerable Domains

A set M is (Dedekind) infinite iff there is an injective function $f\colon M \rightarrow M$ which is not surjective, i.e., with $\mathrm{dom}(f) \neq M$. In first-order logic, we can consider a one-place function symbol f and say that the function $f^{\mathfrak{M}}$ assigned to it in a structure $\mathfrak{M}$ is injective and $\mathrm{ran}(f) \neq |\mathfrak{M}|$:

sol:syn:siz:sec

$$\forall x\,\forall y\,(f(x) = f(y) \rightarrow x = y) \wedge \exists y\,\forall x\, y \neq f(x).$$

If $\mathfrak{M}$ satisfies this sentence, $f^{\mathfrak{M}} : |\mathfrak{M}| \rightarrow |\mathfrak{M}|$ is injective, and so $|\mathfrak{M}|$ must be infinite. If $|\mathfrak{M}|$ is infinite, and hence such a function exists, we can let $f^{\mathfrak{M}}$ be

that function and $\mathfrak{M}$ will satisfy the sentence. However, this requires that our language contains the non-logical symbol f we use for this purpose. In second-order logic, we can simply say that such a function *exists*. This no-longer requires f, and we obtain the sentence in pure second-order logic

$$\text{Inf} \equiv \exists u\, (\forall x\, \forall y\, (u(x) = u(y) \to x = y) \land \exists y\, \forall x\, y \neq u(x)).$$

$\mathfrak{M} \vDash \text{Inf}$ iff $|\mathfrak{M}|$ is infinite. We can then define $\text{Fin} \equiv \neg\text{Inf}$; $\mathfrak{M} \vDash \text{Fin}$ iff $|\mathfrak{M}|$ is finite. No single sentence of pure first-order logic can express that the domain is infinite although an infinite set of them can. There is no set of sentences of pure first-order logic that is satisfied in a structure iff its domain is finite.

Proposition 1.15. $\mathfrak{M} \vDash \text{Inf}$ *iff* $|\mathfrak{M}|$ *is infinite.*

Proof. $\mathfrak{M} \vDash \text{Inf}$ iff $\mathfrak{M}, s \vDash \forall x\, \forall y\, (u(x) = u(y) \to x = y) \land \exists y\, \forall x\, y \neq u(x)$ for some s. If it does, $s(u)$ is an injective function, and some $y \in |\mathfrak{M}|$ is not in the domain of $s(u)$. Conversely, if there is an injective $f\colon |\mathfrak{M}| \to |\mathfrak{M}|$ with $\text{dom}(f) \neq |\mathfrak{M}|$, then $s(u) = f$ is such a variable assignment. □

A set M is enumerable if there is an enumeration

$$m_0, m_1, m_2, \ldots$$

of its elements (without repetitions but possibly finite). Such an enumeration exists iff there is an element $z \in M$ and a function $f\colon M \to M$ such that z, $f(z)$, $f(f(z))$, ..., are all the elements of M. For if the enumeration exists, $z = m_0$ and $f(m_k) = m_{k+1}$ (or $f(m_k) = m_k$ if m_k is the last element of the enumeration) are the requisite element and function. On the other hand, if such a z and f exist, then z, $f(z)$, $f(f(z))$, ..., is an enumeration of M, and M is enumerable. We can express the existence of z and f in second-order logic to produce a sentence true in a structure iff the structure is enumerable:

$$\text{Count} \equiv \exists z\, \exists u\, \forall X\, ((X(z) \land \forall x\, (X(x) \to X(u(x)))) \to \forall x\, X(x))$$

Proposition 1.16. $\mathfrak{M} \vDash \text{Count}$ *iff* $|\mathfrak{M}|$ *is enumerable.*

Proof. Suppose $|\mathfrak{M}|$ is enumerable, and let m_0, m_1, ..., be an enumeration. By removing repetions we can guarantee that no m_k appears twice. Define $f(m_k) = m_{k+1}$ and let $s(z) = m_0$ and $s(u) = f$. We show that

$$\mathfrak{M}, s \vDash \forall X\, ((X(z) \land \forall x\, (X(x) \to X(u(x)))) \to \forall x\, X(x))$$

Suppose $s' \sim_X s$ is arbitrary, and let $M = s'(X)$. Suppose further that $\mathfrak{M}, s' \vDash (X(z) \land \forall x\, (X(x) \to X(u(x))))$. Then $s'(z) \in M$ and whenever $x \in M$, also $s'(u)(x) \in M$. In other words, since $s' \sim_X s$, $m_0 \in M$ and if $x \in M$ then $f(x) \in M$, so $m_0 \in M$, $m_1 = f(m_0) \in M$, $m_2 = f(f(m_0)) \in M$, etc. Thus, $M = |\mathfrak{M}|$, and so $\mathfrak{M} \vDash \forall x\, X(x)s'$. Since s' was an arbitrary X-variant of s, we are done: $\mathfrak{M} \vDash \text{Count}$.

Now assume that $\mathfrak{M} \vDash \text{Count}$, i.e.,

$$\mathfrak{M}, s \vDash \forall X\,((X(z) \land \forall x\,(X(x) \to X(u(x)))) \to \forall x\,X(x))$$

for some s. Let $m = s(z)$ and $f = s(u)$ and consider $M = \{m, f(m), f(f(m)), \dots\}$. Let s' be the X-variant of s with $s(X) = M$. Then

$$\mathfrak{M}, s' \vDash (X(z) \land \forall x\,(X(x) \to X(u(x)))) \to \forall x\,X(x)$$

by assumption. Also, $\mathfrak{M}, s' \vDash X(z)$ since $s'(X) = M \ni m = s'(z)$, and also $\mathfrak{M}, s' \vDash \forall x\,(X(x) \to X(u(x)))$ since whenever $x \in M$ also $f(x) \in M$. So, since both antecedent and conditional are satisfied, the consequent must also be: $\mathfrak{M}, s' \vDash \forall x\,X(x)$. But that means that $M = |\mathfrak{M}|$, and so $|\mathfrak{M}|$ is enumerable since M is, by definition. □

Problem 1.2. The sentence $\text{Inf} \land \text{Count}$ is true in all and only denumerable domains. Adjust the definition of Count so that it becomes a different sentence that directly expresses that the domain is denumerable, and prove that it does.

Chapter 2

Metatheory of Second-order Logic

2.1 Introduction

sol:met:int:sec

First-order logic has a number of nice properties. We know it is not decidable, but at least it is axiomatizable. That is, there are proof systems for first-order logic which are sound and complete, i.e., they give rise to a derivability relation $\vdash$ with the property that for any set of sentences Γ and sentence Q, $\Gamma \vDash \varphi$ iff $\Gamma \vdash \varphi$. This means in particular that the validities of first-order logic are computably enumerable. There is a computable function $f\colon \mathbb{N} \to \mathrm{Sent}(\mathcal{L})$ such that the values of f are all and only the valid sentences of $\mathcal{L}$. This is so because derivations can be enumerated, and those that derive a single sentence are then mapped to that sentence. Second-order logic is more expressive than first-order logic, and so it is in general more complicated to capture its validities. In fact, we'll show that second-order logic is not only undecidable, but its validities are not even computably enumerable. This means there can be no sound and complete proof system for second-order logic (although sound, but incomplete proof systems are available and in fact are important objects of research).

First-order logic also has two more properties: it is compact (if every finite subset of a set Γ of sentences is satisfiable, Γ itself is satisfiable) and the Löwenheim-Skolem Theorem holds for it (if Γ has an infinite model it has a denumerable model). Both of these results fail for second-order logic. Again, the reason is that second-order logic can express facts about the size of domains that first-order logic cannot.

2.2 Second-order Arithmetic

sol:met:spa:sec

Recall that the theory **PA** of Peano arithmetic includes the eight axioms of **Q**,

$$\begin{aligned}
&\forall x\, x' \neq \mathsf{o}\\
&\forall x \forall y\, (x' = y' \to x = y)\\
&\forall x\, (x = \mathsf{o} \vee \exists y\, x = y')\\
&\forall x\, (x + \mathsf{o}) = x\\
&\forall x \forall y\, (x + y') = (x + y)'\\
&\forall x\, (x \times \mathsf{o}) = \mathsf{o}\\
&\forall x \forall y\, (x \times y') = ((x \times y) + x)\\
&\forall x \forall y\, (x < y \leftrightarrow \exists z\, (z' + x) = y)
\end{aligned}$$

plus all sentences of the form

$$(\varphi(\mathsf{o}) \wedge \forall x\, (\varphi(x) \to \varphi(x'))) \to \forall x\, \varphi(x).$$

The latter is a "schema," i.e., a pattern that generates infinitely many sentences of the language of arithmetic, one for each formula $\varphi(x)$. We call this schema the (first-order) *axiom schema of induction*. In *second-order* Peano arithmetic $\mathbf{PA^2}$, induction can be stated as a single sentence. $\mathbf{PA^2}$ consists of the first eight axioms above plus the (second-order) *induction axiom*:

$$\forall X\, (X(\mathsf{o}) \wedge \forall x\, (X(x) \to X(x'))) \to \forall x\, X(x)).$$

It says that if a subset X of the domain contains $\mathsf{o}^{\mathfrak{M}}$ and with any $x \in |\mathfrak{M}|$ also contains $\prime^{\mathfrak{M}}(x)$ (i.e., it is "closed under successor") it contains everything in the domain (i.e., $X = |\mathfrak{M}|$).

The induction axiom guarantees that any structure satisfying it contains only those elements of $|\mathfrak{M}|$ the axioms require to be there, i.e., the values of $\overline{n}$ for $n \in \mathbb{N}$. A model of $\mathbf{PA^2}$ contains no non-standard numbers.

Theorem 2.1. *If* $\mathfrak{M} \vDash \mathbf{PA^2}$ *then* $|\mathfrak{M}| = \{\mathrm{Val}^{\mathfrak{M}}(\overline{n}) : n \in \mathbb{N}\}$.

sol:met:spa: thm:sol-pa-standard

Proof. Let $N = \{\mathrm{Val}^{\mathfrak{M}}(\overline{n}) : n \in \mathbb{N}\}$, and suppose $\mathfrak{M} \vDash \mathbf{PA^2}$. Of course, for any $n \in \mathbb{N}$, $\mathrm{Val}^{\mathfrak{M}}(\overline{n}) \in |\mathfrak{M}|$, so $N \subseteq |\mathfrak{M}|$.

Now for inclusion in the other direction. Consider a variable assignment s with $s(X) = N$. By assumption,

$$\begin{aligned}
&\mathfrak{M} \vDash \forall X\, (X(\mathsf{o}) \wedge \forall x\, (X(x) \to X(x'))) \to \forall x\, X(x), \text{ thus}\\
&\mathfrak{M}, s \vDash (X(\mathsf{o}) \wedge \forall x\, (X(x) \to X(x'))) \to \forall x\, X(x).
\end{aligned}$$

Consider the antecedent of this conditional. $\mathrm{Val}^{\mathfrak{M}}(\mathsf{o}) \in N$, and so $\mathfrak{M}, s \vDash X(\mathsf{o})$. The second conjunct, $\forall x\, (X(x) \to X(x'))$ is also satisfied. For suppose $x \in N$. By definition of N, $x = \mathrm{Val}^{\mathfrak{M}}(\overline{n})$ for some n. That gives $\prime^{\mathfrak{M}}(x) = \mathrm{Val}^{\mathfrak{M}}(\overline{n+1}) \in N$. So, $\prime^{\mathfrak{M}}(x) \in N$.

We have that $\mathfrak{M}, s \vDash X(\mathsf{o}) \wedge \forall x\, (X(x) \to X(x'))$. Consequently, $\mathfrak{M}, s \vDash \forall x\, X(x)$. But that means that for every $x \in |\mathfrak{M}|$ we have $x \in s(X) = N$. So, $|\mathfrak{M}| \subseteq N$. □

sol:met:spa:
cor:sol-pa-categorical

Corollary 2.2. *Any two models of* $\mathbf{PA^2}$ *are isomorphic.*

Proof. By Theorem 2.1, the domain of any model of $\mathbf{PA^2}$ is exhausted by $\mathrm{Val}^{\mathfrak{M}}(\overline{n})$. Any such model is also a model of $\mathbf{Q}$. By **??**, any such model is standard, i.e., isomorphic to $\mathfrak{N}$. □

Above we defined $\mathbf{PA^2}$ as the theory that contains the first eight arithmetical axioms plus the second-order induction axiom. In fact, thanks to the expressive power of second-order logic, only the *first two* of the arithmetical axioms plus induction are needed for second-order Peano arithmetic.

sol:met:spa:
prop:sol-pa-definable

Proposition 2.3. *Let* $\mathbf{PA^{2\dagger}}$ *be the second-order theory containing the first two arithmetical axioms (the successor axioms) and the second-order induction axiom. Then* $\leq$, $+$, *and* $\times$ *are definable in* $\mathbf{PA^{2\dagger}}$.

Proof. To show that $\leq$ is definable, we have to find a formula $\varphi_{\leq}(x, y)$ such that $\mathfrak{N} \vDash \varphi(\overline{n}, \overline{m})$ iff $n < m$. Consider the formula

$$\psi(x, Y) \equiv Y(x) \wedge \forall y\,(Y(y) \rightarrow Y(y'))$$

Clearly, $\psi(\overline{n}, Y)$ is satisfied by a set $Y \subseteq \mathbb{N}$ iff $\{m : n \leq m\} \subseteq Y$, so we can take $\varphi_{\leq}(x, y) \equiv \forall Y\,(\psi(x, Y) \rightarrow Y(y))$. □

Problem 2.1. Complete the proof of Proposition 2.3.

Corollary 2.4. $\mathfrak{M} \vDash \mathbf{PA^2}$ *iff* $\mathfrak{M} \vDash \mathbf{PA^{2\dagger}}$.

Proof. Immediate from Proposition 2.3. □

2.3 Second-order Logic is not Axiomatizable

sol:met:nax:
sec
sol:met:nax:
thm:sol-undecidable

Theorem 2.5. *Second-order logic is undecidable.*

Proof. A first-order sentence is valid in first-order logic iff it is valid in second-order logic, and first-order logic is undecidable. □

sol:met:nax:
cor:sol-not-axiomatizable

Theorem 2.6. *There is no sound and complete proof system for second-order logic.*

Proof. Let φ be a sentence in the language of arihmetic. $\mathfrak{N} \vDash \varphi$ iff $\mathbf{PA^2} \vDash \varphi$. Let P be the conjunction of the nine axioms of $\mathbf{PA^2}$. $\mathbf{PA^2} \vDash \varphi$ iff $\vDash P \rightarrow \varphi$, i.e., $\mathfrak{M} \vDash P \rightarrow \varphi$. Now consider the sentence $\forall z\,\forall u\,\forall u'\,\forall u''\,\forall L\,(P' \rightarrow \varphi')$ resulting by replacing o by z, $'$ by the one-place function variable u, $+$ and $\times$ by the two-place function-variables u' and u'', respectively, and $<$ by the two-place relation variable L and universally quantifying. It is a valid sentence of pure second-order logic iff the original sentence was valid iff $\mathbf{PA^2} \vDash \varphi$ iff $\mathfrak{N} \vDash \varphi$. Thus if there were a sound and complete proof system for second-order logic,

we could use it to define a computable enumeration $f : \mathbb{N} \to \text{Sent}(\mathcal{L}_A)$ of the sentences true in $\mathfrak{N}$. This function would be representable in $\mathbf{Q}$ by some first-order formula $\psi_f(x, y)$. Then the formula $\exists x\, \psi_f(x, y)$ would define the set of true first-order sentences of $\mathfrak{N}$, contradicting Tarski's Theorem. □

2.4 Second-order Logic is not Compact

explanation

Call a set of sentences Γ *finitely satisfiable* if every one of its finite subsets is satisfiable. First-order logic has the property that if a set of sentences Γ is finitely satisfiable, it is satisfiable. This property is called *compactness*. It has an equivalent version involving entailment: if $\Gamma \vDash \varphi$, then already $\Gamma_0 \vDash \varphi$ for some finite subset $\Gamma_0 \subseteq \Gamma$. In this version it is an immediate corollary of the completeness theorem: for if $\Gamma \vDash \varphi$, by completeness $\Gamma \vdash \varphi$. But a derivation can only make use of finitely many sentences of Γ.

sol:met:com:sec

Compactness is not true for second-order logic. There are sets of second-order sentences that are finitely satisfiable but not satisfiable, and that entail some φ without a finite subset entailing φ.

Theorem 2.7. *Second-order logic is not compact.*

sol:met:com:thm:sol-undecidable

Proof. Recall that

$$\text{Inf} \equiv \exists u\, (\forall x\, \forall y\, (u(x) = u(y) \to x = y) \wedge \exists y\, \forall x\, y \neq u(x))$$

is satisfied in a structure iff its domain is infinite. Let $\varphi^{\geq n}$ be a sentence that asserts that the domain has at least n elements, e.g.,

$$\varphi^{\geq n} \equiv \exists x_1 \ldots \exists x_n\, (x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \cdots \wedge x_{n-1} \neq x_n).$$

Consider the set of sentences

$$\Gamma = \{\neg\text{Inf}, \varphi^{\geq 1}, \varphi^{\geq 2}, \varphi^{\geq 3}, \ldots\}.$$

It is finitely satisfiable, since for any finite subset $\Gamma_0 \subseteq \Gamma$ there is some k so that $\varphi^{\geq k} \in \Gamma$ but no $\varphi^{\geq n} \in \Gamma$ for $n > k$. If $|\mathfrak{M}|$ has k elements, $\mathfrak{M} \vDash \Gamma_0$. But, Γ is not satisfiable: if $\mathfrak{M} \vDash \neg\text{Inf}$, $|\mathfrak{M}|$ must be finite, say, of size k. Then $\mathfrak{M} \nvDash \varphi^{\geq k+1}$. □

Problem 2.2. Give an example of a set Γ and a sentence φ so that $\Gamma \vDash \varphi$ but for every finite subset $\Gamma_0 \subseteq \Gamma$, $\Gamma_0 \nvDash \varphi$.

2.5 The Löwenheim-Skolem Theorem Fails for Second-order Logic

explanation

The (Downward) Löwenheim-Skolem Theorem states that every set of sentences with an infinite model has an enumerable model. It, too, is a consequence of the completeness theorem: the proof of completeness generates a

sol:met:lst:sec

model for any consistent set of sentences, and that model is enumerable. There is also an Upward Löwenheim-Skolem Theorem, which guarantees that if a set of sentences has a denumerable model it also has a non-enumerable model. Both theorems fail in second-order logic.

sol:met:lst:thm:sol-no-ls

Theorem 2.8. *The Löwenheim-Skolem Theorem fails for second-order logic: There are sentences with infinite models but no enumerable models.*

Proof. Recall that

$$\text{Count} \equiv \exists z\, \exists u\, \forall X\, ((X(z) \wedge \forall x\, (X(x) \to X(u(x)))) \to \forall x\, X(x))$$

is true in a structure $\mathfrak{M}$ iff $|\mathfrak{M}|$ is enumerable, so $\neg$Count is true in $\mathfrak{M}$ iff $|\mathfrak{M}|$ is non-enumerable. There are such structures—take any non-enumerable set as the domain, e.g., $\wp(\mathbb{N})$ or $\mathbb{R}$. So $\neg$Count has infinite models but no enumerable models. □

Theorem 2.9. *There are sentences with denumerable but no non-enumerable models.*

Proof. Count $\wedge$ Inf is true in $\mathbb{N}$ but not in any structure $\mathfrak{M}$ with $|\mathfrak{M}|$ non-enumerable. □

Chapter 3

Second-order Logic and Set Theory

This section deals with coding powersets and the continuum in second-order logic. The results are stated but proofs have yet to be filled in. There are no problems yet—and the definitions and results themselves may have problems. Use with caution and report anything that's false or unclear.

3.1 Introduction

sol:set:int:sec

Since second-order logic can quantify over subsets of the domain as well as functions, it is to be expected that some amount, at least, of set theory can be carried out in second-order logic. By "carry out," we mean that it is possible to express set theoretic properties and statements in second-order logic, and is possible without any special, non-logical vocabulary for sets (e.g., the membership predicate symbol of set theory). For instance, we can define unions and intersections of sets and the subset relationship, but also compare the sizes of sets, and state results such as Cantor's Theorem.

3.2 Comparing Sets

sol:set:cmp:sec

Proposition 3.1. *The formula* $\forall x\,(X(x) \to Y(x))$ *defines the subset relation, i.e.,* $\mathfrak{M}, s \vDash \forall x\,(X(x) \to Y(x))$ *iff* $s(X) \subseteq S(y)$.

Proposition 3.2. *The formula* $\forall x\,(X(x) \leftrightarrow Y(x))$ *defines the identity relation on sets, i.e.,* $\mathfrak{M}, s \vDash \forall x\,(X(x) \leftrightarrow Y(x))$ *iff* $s(X) = S(y)$.

Proposition 3.3. *The formula* $\exists x\, X(x)$ *defines the property of being non-empty, i.e.,* $\mathfrak{M}, s \vDash \exists x\, X(x)$ *iff* $s(X) \neq \emptyset$.

A set X is no larger than a set Y, $X \preceq Y$, iff there is an injective function $f\colon X \to Y$. Since we can express that a function is injective, and also that its values for arguments in X are in Y, we can also define the relation of being no larger than on subsets of the domain.

Proposition 3.4. *The formula*

$$\exists u\,(\forall x\,(X(x) \to Y(u(x))) \wedge \forall x\,\forall y\,(u(x) = u(y) \to x = y))$$

defines the relation of being no larger than.

Two sets are the same size, or "equinumerous," $X \approx Y$, iff there is a bijective function $f\colon X \to Y$.

Proposition 3.5. *The formula*

$$\begin{aligned}\exists u\,(\forall x\,(X(x) \to Y(u(x))) \wedge{}& \\ \forall x\,\forall y\,(u(x) = u(y) \to x = y) \wedge{}& \\ \forall y\,(Y(y) \to \exists x\,(X(x) \wedge y = u(x))))&\end{aligned}$$

defines the relation of being equinumerous with.

We will abbreviate these formulas, respectively, as $X \subseteq Y$, $X = Y$, $X \neq \emptyset$, $X \preceq Y$, and $X \approx Y$. (This may be slightly confusing, since we use the same notation when we speak informally about sets X and Y—but here the notation is an abbreviation for formulas in second-order logic involving one-place relation variables X and Y.)

Proposition 3.6. *The sentence* $\forall X\,\forall Y\,((X \preceq Y \wedge Y \preceq X) \to X \approx Y)$ *is valid.*

Proof. The sentence is satisfied in a structure $\mathfrak{M}$ if, for any subsets $X \subseteq |\mathfrak{M}|$ and $Y \subseteq |\mathfrak{M}|$, if $X \preceq Y$ and $Y \preceq X$ then $X \approx Y$. But this holds for *any* sets X and Y—it is the Schröder-Bernstein Theorem. □

3.3 Cardinalities of Sets

sol:set:crd: sec

Just as we can express that the domain is finite or infinite, enumerable or non-enumerable, we can define the property of a subset of $|\mathfrak{M}|$ being finite or infinite, enumerable or non-enumerable.

explanation

Proposition 3.7. *The formula* $\mathrm{Inf}(X) \equiv$

$$\begin{aligned}\exists u\,(\forall x\,\forall y\,(u(x) = u(y) \to x = y) \wedge{}& \\ \exists y\,(X(y) \wedge \forall x\,(X(x) \to y \neq u(x)))&\end{aligned}$$

is satisfied with respect to a variable assignment s iff $s(X)$ is infinite.

Proposition 3.8. *The formula* $\mathrm{Count}(X) \equiv$

$$\exists z\, \exists u\, (X(z) \wedge \forall x\, (X(x) \to X(u(x))) \wedge \forall Y\, ((Y(z) \wedge \forall x\, (Y(x) \to Y(u(x)))) \to X = Y))$$

is satisfied with respect to a variable assignment s *iff* $s(X)$ *is enumerable*

We know from Cantor's Theorem that there are non-enumerable sets, and in fact, that there are infinitely many different levels of infinite sizes. Set theory develops an entire arithmetic of sizes of sets, and assigns infinite cardinal numbers to sets. The natural numbers serve as the cardinal numbers measuring the sizes of finite sets. The cardinality of denumerable sets is the first infinite cardinality, called $\aleph_0$ ("aleph-nought" or "aleph-zero"). The next infinite size is $\aleph_1$. It is the smallest size a set can be without being countable (i.e., of size $\aleph_0$). We can define "X has size $\aleph_0$" as $\mathrm{Aleph}_0(X) \leftrightarrow \mathrm{Inf}(X) \wedge \mathrm{Count}(X)$. X has size $\aleph_1$ iff all its subsets are finite or have size $\aleph_0$, but is not itself of size $\aleph_0$. Hence we can express this by the formula $\mathrm{Aleph}_1(X) \equiv \forall Y\, (Y \subseteq X \to (\neg \mathrm{Inf}(Y) \vee \mathrm{Aleph}_0(Y))) \wedge \neg \mathrm{Aleph}_0(X)$. Being of size $\aleph_2$ is defined similarly, etc.

There is one size of special interest, the so-called cardinality of the continuum. It is the size of $\wp(\mathbb{N})$, or, equivalently, the size of $\mathbb{R}$. That a set is the size of the continuum can also be expressed in second-order logic, but requires a bit more work.

3.4 The Power of the Continuum

explanation

In second-order logic we can quantify over subsets of the domain, but not over sets of subsets of the domain. To do this directly, we would need *third-order* logic. For instance, if we wanted to state Cantor's Theorem that there is no injective function from the power set of a set to the set itself, we might try to formulate it as "for every set X, and every set P, if P is the power set of X, then not $P \preceq X$. And to say that P is the power set of X would require formalizing that the elements of P are all and only the subsets of X, so something like $\forall Y\, (P(Y) \leftrightarrow Y \subseteq X)$. The problem lies in $P(Y)$: that is not a formula of second-order logic, since only terms can be arguments to one-place relation variables like P.

sol:set:pow:sec

We can, however, *simulate* quantification over sets of sets, if the domain is large enough. The idea is to make use of the fact that two-place relations R relates elements of the domain to elements of the domain. Given such an R, we can collect all the elements to which some x is R-related: $\{y \in |\mathfrak{M}| : R(x, y)\}$ is the set "coded by" x. Converseley, if $Z \subseteq \wp(|\mathfrak{M}|)$ is some collection of subsets of $|\mathfrak{M}|$, and there are at least as many elements of $|\mathfrak{M}|$ as there are sets in Z, then there is also a relation $R \subseteq |\mathfrak{M}|^2$ such that every $Y \in Z$ is coded by some x using R.

Definition 3.9. If $R \subseteq |\mathfrak{M}|^2$, then x *R-codes* $\{y \in |\mathfrak{M}| : R(x, y)\}$. Y R-codes $\wp(X)$ iff for every $Z \subseteq X$, some $x \in Y$ R-codes Y, and every $x \in Y$ R-codes some $Y \in Z$.

Proposition 3.10. *The formula*

$$\mathrm{Codes}(x, R, Y) \equiv \forall y\,(Y(y) \leftrightarrow R(x, y))$$

expresses that $s(x)$ $s(R)$-codes $s(Y)$. The formula

$$\begin{aligned}\mathrm{Pow}(Y, R, X) \equiv \\ \forall Z\,(Z \subseteq X \to \exists x\,(Y(x) \wedge \mathrm{Codes}(x, R, Z))) \wedge \\ \forall x\,(Y(x) \to \forall Z\,(\mathrm{Codes}(x, R, Z) \to Z \subseteq X)\end{aligned}$$

expresses that $s(Y)$ $s(R)$-codes the power set of $s(X)$.

With this trick, we can express statements about the power set by quantifying over the codes of subsets rather than the subsets themselves. For instance, Cantor's Theorem can now be expressed by saying that there is no injective function from the domain of any relation that codes the power set of X to X itself. explanation

Proposition 3.11. *The sentence*

$$\begin{aligned}\forall X\,\forall R\,(\mathrm{Pow}(R, X) \to \\ \neg\exists u\,(\forall x\,\forall y\,(u(x) = u(y) \to x = y) \wedge \\ \forall Y\,(\mathrm{Codes}(x, R, Y) \to X(u(x)))))\end{aligned}$$

is valid.

The power set of a denumerable set is non-enumerable, and so its cardinality is larger than that of any denumerable set (which is $\aleph_0$). The size of $\wp(\mathbb{R})$ is called the "power of the continuum," since it is the same size as the points on the real number line, $\mathbb{R}$. If the domain is large enough to code the power set of a denumerable set, we can express that a set is the size of the continuum by saying that it is equinumerous with any set Y that codes the power set of set X of size $\aleph_0$. (If the domain is not large enough, i.e., it contains no subset equinumerous with $\mathbb{R}$, then there can also be no relation that codes $\wp(X)$.) explanation

Proposition 3.12. *If $\mathbb{R} \preceq |\mathfrak{M}|$, then the formula*

$$\mathrm{Cont}(X) \equiv \forall X\,\forall Y\,\forall R\,((\mathrm{Aleph}_0(X) \wedge \mathrm{Pow}(Y, R, X)) \to \neg Y \preceq X)$$

expresses that $s(X) \approx \mathbb{R}$.

Proposition 3.13. *$|\mathfrak{M}| \approx \mathbb{R}$ iff*

$$\begin{aligned}\mathfrak{M} \vDash \exists X\,\exists Y\,\exists R\,(\mathrm{Aleph}_0(X) \wedge \mathrm{Pow}(Y, R, X) \wedge \\ \exists u\,(\forall x\,\forall y\,(u(x) = u(y) \to x = y) \wedge \\ \forall y\,(Y(y) \to \exists x\, y = u(x)))).\end{aligned}$$

explanation The Continuum Hypothesis is the statement that the size of the continuum is the first non-enumerable cardinality, i.e, that $\wp(\mathbb{N})$ has size $\aleph_1$.

Proposition 3.14. *The Continuum Hypothesis is true iff*

$$\mathrm{CH} \equiv \forall X\,(\mathrm{Aleph}_1(X) \leftrightarrow \mathrm{Cont}(x))$$

is valid.

Note that it isn't true that $\neg$CH is valid iff the Continuum Hypothesis is false. In an enumerable domain, there are no subsets of size $\aleph_1$ and also no subsets of the size of the continuum, so CH is always true in an enumerable domain. However, we can give a different sentence that is valid iff the Continuum Hypothesis is false:

Proposition 3.15. *The Continuum Hypothesis is false iff*

$$\mathrm{NCH} \equiv \forall X\,(\mathrm{Cont}(X) \rightarrow \exists Y\,(Y \subseteq X \wedge \neg\mathrm{Count}(X) \wedge \neg X \approx Y))$$

is valid.

Photo Credits

Bibliography