Chapter udf

Syntax and Semantics

syn.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: $\mathfrak{M}, s \models \varphi$ 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 $\mathfrak{M}, s \models \varphi$, 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 $\forall v_0\, (P^1_0(v_0) \lor \lnot P^1_0(v_0))$ is a formula in both first- and second-order logic, in the latter we can also consider $\forall V^1_0\, \forall v_0\, (V^1_0(v_0) \lor \lnot V^1_0(v_0))$ and $\exists V^1_0\, \forall v_0\, (V^1_0(v_0) \lor \lnot V^1_0(v_0))$. 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 $\forall v_0\, (V^1_0(v_0) \lor \lnot V^1_0(v_0))$ holds for all ways of assigning a subset of $|\mathfrak{M}|$ as the value of V_0^1, or for at least one. Since every set either contains or fails to contain a given object, both are true in any structure.

syn.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_0^n, V_1^n, V_2^n, …
2. A denumerable set of second-order function variables: u_0^n, u_1^n, u_2^n, …

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_i^n and u, v, etc., as meta-variables for u_i^n.

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 syn.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 $t_1, \dots, t_n$ are terms, then $u(t_1, \dots, t_n)$ is a term.

explanation 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.

Definition syn.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 $t_1, \dots, t_n$ are second-order terms of $\mathcal{L}$, then $X(t_1, \dots, t_n)$ 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.

syn.3 Satisfaction

sol:syn:sat:sec To define the satisfaction relation $\mathfrak{M}, s \models \varphi$ for second-order formulas, we have to extend the definitions to cover second-order variables. explanation

Definition syn.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 very term. explanation

Definition syn.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:

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

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

Definition syn.5 (Satisfaction)**.** For second-order formulas φ, the definition of satisfaction is like **??** with the addition of:

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

Example syn.6. $\mathfrak{M}, s \vDash \forall z\,(Xz \leftrightarrow \neg Yz)$ whenever $s(Y) = |\mathfrak{M}| \setminus s(X)$. So for instance, let $|\mathfrak{M}| = \{1, 2, 3\}$, $s(X) = \{1, 2\}$ and $s(Y) = \{3\}$.

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

syn.4 Semantic Notions

explanation 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. sol:syn:sem:sec

Definition syn.7 (Validity)**.** A sentence φ is *valid*, $\vDash \varphi$, iff $\mathfrak{M} \vDash \varphi$ for every structure $\mathfrak{M}$.

Definition syn.8 (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 syn.9 (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*.

syn.5 Expressive Power

explanation 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 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. sol:syn:exp:sec

Example syn.10. 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 $\varphi_R(v_0, v_1)$ with only the variables v_0 and v_1 free, such that $R(x, y)$ holds (i.e., $\langle x, y\rangle \in R$) iff $\mathfrak{M}, s \vDash \varphi_R(v_0, v_1)$ for $s(v_0) = x$ and $s(v_1) = y$. For instance, in first-order logic we can define the identity relation $\mathrm{Id}_{|\mathfrak{M}|}$ (i.e., $\{\langle x, x\rangle : x \in |\mathfrak{M}|\}$) by the formula $v_0 = v_1$. In second-order logic, we can define this relation *without* $=$. For if x and y 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 x and y are different, then they are not elements of the same subsets: e.g., $x \in \{x\}$ but $y \notin \{x\}$ if $x \neq y$. So "being elements of the same subsets of $|\mathfrak{M}|$" is a relation that holds of x and y iff $x = y$. 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 $\mathrm{Id}_{|\mathfrak{M}|}$:

$$\forall X\,(X(\mathsf{v}_0) \leftrightarrow X(\mathsf{v}_1))$$

Problem syn.1. Show that $\forall X\,(X(\mathsf{v}_0) \to X(\mathsf{v}_1))$ (note: $\to$ not $\leftrightarrow$!) defines $\mathrm{Id}_{|\mathfrak{M}|}$.

Example syn.11. If R is a two-place predicate symbol, $R^{\mathfrak{M}}$ is a two-place relation on $|\mathfrak{M}|$. Its *transitive closure* R^* is the relation that holds between x and y if for some $z_1, \ldots, z_k$, $R(x, z_1)$, $R(z_1, z_2)$, $\ldots$, $R(z_k, y)$ holds. This includes the case if $k = 0$, i.e., if $R(x, y)$ holds. 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:

$$\psi_R(X) \equiv \forall x\,\forall y\,(R(x, y) \to X(x, y)) \land \\ \forall x\,\forall y\,\forall z\,((X(x, y) \land X(y, z)) \to X(x, z))$$

Here, somewhat confusingly, we use R as the predicate symbol for R. 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

$$R^*(X) \equiv \psi_R(X) \land \forall Y\,(\psi_R(Y) \to \forall x\,\forall y\,(X(x, y) \to Y(x, y)))$$

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

syn.6 Describing Infinite and Enumerable Domains

sol:syn:siz:sec A set M is (Dedekind) infinite iff there is an injective function $f\colon M \to M$ 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 $\mathrm{ran}(f) \neq |\mathfrak{M}|$:

$$\forall x\,\forall y\,(f(x) = f(y) \to x = y) \land \exists y\,\forall x\,y \neq f(x)$$

If $\mathfrak{M}$ satisfies this sentence, $f^{\mathfrak{M}} : |\mathfrak{M}| \to |\mathfrak{M}|$ 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 have the sentence in pure second-order logic

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

$\mathfrak{M} \models \mathrm{Inf}$ iff $|\mathfrak{M}|$ is infinite. We can then define $\mathrm{Fin} \equiv \lnot\mathrm{Inf}$; $\mathfrak{M} \models \mathrm{Fin}$ 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 syn.12. $\mathfrak{M} \models \mathrm{Inf}$ *iff* $|\mathfrak{M}|$ *is infinite.*

Proof. $\mathfrak{M} \models \mathrm{Inf}$ iff $\mathfrak{M}, s \models \forall x\,\forall y\,(u(x) = u(y) \to x = y) \land \exists y\,\forall x\, y \neq u(x)$ 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 $\mathrm{dom}(f) \neq |\mathfrak{M}|$, then $s(u) = f$ is such a variable assignment. □

A set M is enumerable if there is an enumeration

$$m_0, m_1, m_2, \dots$$

of its elements (without repetitions). 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:

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

Proposition syn.13. $\mathfrak{M} \models \mathrm{Count}$ *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

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

Suppose $s' \sim_X s$, and $M = s'(X)$. Suppose further that $\mathfrak{M}, s' \models (X(z) \land \forall x\,(X(x) \to X(u(x))))$. 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 $\mathfrak{M} \models \forall x\, X(x)s'$. Since s' was an arbitrary X-variant of s, we are done.

Now assume that

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

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

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

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

Problem syn.2. The sentence $\mathrm{Inf} \land \mathrm{Count}$ 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.

Photo Credits

Bibliography