

Chapter udf

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.

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 $\forall v_0 (P_0^1(v_0) \vee \neg P_0^1(v_0))$ is a formula in both first- and second-order logic, in the latter we can also consider $\forall V_0^1 \forall v_0 (V_0^1(v_0) \vee \neg V_0^1(v_0))$ and $\exists V_0^1 \forall v_0 (V_0^1(v_0) \vee \neg V_0^1(v_0))$. Since these contain no free variables, they are sentences of second-order logic. Here, V_0^1 is a second-order 1-place predicate variable. The allowable interpretations of V_0^1 are the same that we can assign to a 1-place predicate symbol like P_0^1 , i.e., subsets of $|\mathfrak{M}|$. Quantification over them then amounts to saying that $\forall v_0 (V_0^1(v_0) \vee \neg V_0^1(v_0))$ 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

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 : $V_0^n, V_1^n, V_2^n, \dots$
2. A denumerable set of second-order function variables: $u_0^n, u_1^n, u_2^n, \dots$

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, w , 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} , $\text{Trm}^2(\mathcal{L})$, 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.

So, a second-order term looks just like a first-order term, except that where [explanation](#) 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* $\text{Frm}^2(\mathcal{L})$ 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 [explanation](#) 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 difference for variable assignments s : these now must not just provide values for the first-order **variables**, but also for the second-order **variables**.

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., $s(u_i^n): |\mathfrak{M}|^n \rightarrow |\mathfrak{M}|$;

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

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** $\text{Val}_s^{\mathfrak{M}}(t)$ is defined as for first-order terms, plus the following clause:

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

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

Definition syn.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 syn.6. If s is a variable assignment for a structure \mathfrak{M} and $m \in |\mathfrak{M}|$, then the assignment $s[m/x]$ is the variable assignment defined by

$$s[m/y] = \begin{cases} m & \text{if } y \equiv x \\ s(y) & \text{otherwise,} \end{cases}$$

If X is an n -place relation variable and $M \subseteq |\mathfrak{M}|^n$, then $s[M/X]$ is the variable assignment defined by

$$s[M/y] = \begin{cases} M & \text{if } y \equiv X \\ s(y) & \text{otherwise.} \end{cases}$$

If u is an n -place function variable and $f: |\mathfrak{M}|^n \rightarrow |\mathfrak{M}|$, then $s[f/u]$ is the variable assignment defined by

$$s[f/y] = \begin{cases} f & \text{if } y \equiv u \\ s(y) & \text{otherwise.} \end{cases}$$

In each case, y may be any first- or second-order variable.

Definition syn.7 (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 $\langle \text{Val}_s^{\mathfrak{M}}(t_1), \dots, \text{Val}_s^{\mathfrak{M}}(t_n) \rangle \in s(X^n)$.
2. $\varphi \equiv \forall X \psi$: $\mathfrak{M}, s \models \varphi$ iff for every $M \subseteq |\mathfrak{M}|^n$, $\mathfrak{M}, s[M/X] \models \psi$.
3. $\varphi \equiv \exists X \psi$: $\mathfrak{M}, s \models \varphi$ iff for at least one $M \subseteq |\mathfrak{M}|^n$ so that $\mathfrak{M}, s[M/X] \models \psi$.
4. $\varphi \equiv \forall u \psi$: $\mathfrak{M}, s \models \varphi$ iff for every $f: |\mathfrak{M}|^n \rightarrow |\mathfrak{M}|$, $\mathfrak{M}, s[f/u] \models \psi$.
5. $\varphi \equiv \exists u \psi$: $\mathfrak{M}, s \models \varphi$ iff for at least one $f: |\mathfrak{M}|^n \rightarrow |\mathfrak{M}|$ so that $\mathfrak{M}, s[f/u] \models \psi$.

Example syn.8. 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 (it includes free variables X and Y), it is only satisfied relative to a structure \mathfrak{M} together with a variable assignment s .

$\mathfrak{M}, s \models \forall z (X(z) \leftrightarrow \neg Y(z))$ whenever the elements of $s(X)$ are not elements of $s(Y)$, and vice versa, i.e., iff $s(Y) = |\mathfrak{M}| \setminus s(X)$. 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 $\mathfrak{M}, s_1 \models \forall z (X(z) \leftrightarrow \neg Y(z))$.

By contrast, if we have $s_2(X) = \{1, 2\}$ and $s_2(Y) = \{2, 3\}$, $\mathfrak{M}, s_2 \not\models \forall z (X(z) \leftrightarrow \neg Y(z))$. That's because $\mathfrak{M}, s_2[2/z] \models X(z)$ (since $2 \in s_2[2/z](X)$) but $\mathfrak{M}, s_2[2/z] \not\models \neg Y(z)$ (since also $2 \in s_2[2/z](Y)$).

Example syn.9. $\mathfrak{M}, s \models \exists Y (\exists y Y(y) \wedge \forall z (X(z) \leftrightarrow \neg Y(z)))$ if there is an $N \subseteq |\mathfrak{M}|$ such that $\mathfrak{M}, s[N/Y] \models (\exists y Y(y) \wedge \forall z (X(z) \leftrightarrow \neg Y(z)))$. And that is the case for any $N \neq \emptyset$ (so that $\mathfrak{M}, s[N/Y] \models \exists y Y(y)$) and, as in the previous example, $M = |\mathfrak{M}| \setminus s(X)$. In other words, $\mathfrak{M}, s \models \exists Y (\exists y Y(y) \wedge \forall z (X(z) \leftrightarrow \neg Y(z)))$ 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

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

is satisfied relative to any assignment s , since we can always find $M \subseteq |\mathfrak{M}|$ but $M \neq |\mathfrak{M}|$ (e.g., $M = \emptyset$).

Example syn.10. The second-order sentence $\forall X \forall y X(y)$ says that every 1-place relation, i.e., every property, holds of every object. That is clearly never true, since in every \mathfrak{M} , for a variable assignment s with $s(X) = \emptyset$, and $s(y) = a \in |\mathfrak{M}|$ we have $\mathfrak{M}, s \not\models X(y)$. This means that $\varphi \rightarrow \forall X \forall y X(y)$ is equivalent in second-order logic to $\neg\varphi$, that is: $\mathfrak{M} \models \varphi \rightarrow \forall X \forall y X(y)$ iff $\mathfrak{M} \models \neg\varphi$. In other words, in second-order logic we can define \neg using \forall and \rightarrow .

Problem syn.1. Show that in second-order logic \forall and \rightarrow can define the other connectives:

1. Prove that in second-order logic $\varphi \wedge \psi$ is equivalent to $\forall X (\varphi \rightarrow (\psi \rightarrow \forall x X(x)) \rightarrow \forall x X(x))$.
2. Find a second-order formula using only \forall and \rightarrow equivalent to $\varphi \vee \psi$.

syn.4 Semantic Notions

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.

Definition syn.11 (Validity). A sentence φ is *valid*, $\models \varphi$, iff $\mathfrak{M} \models \varphi$ for every **structure** \mathfrak{M} .

Definition syn.12 (Entailment). A set of sentences Γ *entails* a sentence φ , $\Gamma \models \varphi$, iff for every **structure** \mathfrak{M} with $\mathfrak{M} \models \Gamma$, $\mathfrak{M} \models \varphi$.

Definition syn.13 (Satisfiability). A set of sentences Γ is *satisfiable* if $\mathfrak{M} \models \Gamma$ 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 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.

sol:syn:exp:
sec

Definition syn.14. 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 $\mathfrak{M}, s \models \varphi_R(x, y)$ for $s(x) = a$ and $s(y) = b$.

Example syn.15. In first-order logic we can define the identity relation $\text{Id}_{|\mathfrak{M}|}$ (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 $\text{Id}_{|\mathfrak{M}|}$:

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

Problem syn.2. Show that $\forall X (X(x) \rightarrow X(y))$ (note: \rightarrow not $\leftrightarrow!$) defines $\text{Id}_{|\mathfrak{M}|}$.

Example syn.16. 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, \dots, c_k , $R(a, c_1)$, $R(c_1, c_2)$, \dots , $R(c_k, b)$ 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 :

$$\begin{aligned}\psi_R(X) \equiv & \forall x \forall y (R(x, y) \rightarrow X(x, y)) \wedge \\ & \forall x \forall y \forall z ((X(x, y) \wedge X(y, z)) \rightarrow X(x, z)).\end{aligned}$$

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 $\mathfrak{M}, s \models R^*(X)$ 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 $f: M \rightarrow M$ which is not surjective, i.e., with $\text{dom}(f) \neq M$. 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 $\text{ran}(f) \neq |\mathfrak{M}|$:

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

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

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

$\mathfrak{M} \models \text{Inf}$ iff $|\mathfrak{M}|$ is infinite. We can then define $\text{Fin} \equiv \neg \text{Inf}$; $\mathfrak{M} \models \text{Fin}$ 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 syn.17. $\mathfrak{M} \models \text{Inf}$ iff $|\mathfrak{M}|$ is infinite.

Proof. $\mathfrak{M} \models \text{Inf}$ iff $\mathfrak{M}, s \models \forall x \forall y (u(x) = u(y) \rightarrow x = y) \wedge \exists y \forall x y \neq u(x)$ 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 $f: |\mathfrak{M}| \rightarrow |\mathfrak{M}|$ with $\text{dom}(f) \neq |\mathfrak{M}|$, then $s(u) = f$ is such a variable assignment. \square

A set M is enumerable if there is an enumeration

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

of its elements (without repetitions but possibly finite). Such an enumeration exists iff there is an element $z \in M$ and a function $f: M \rightarrow M$ such that $z, f(z), f(f(z)), \dots$, 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 $z, f(z), f(f(z)), \dots$, 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:

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

Proposition syn.18. $\mathfrak{M} \models \text{Count}$ iff $|\mathfrak{M}|$ is enumerable.

Proof. Suppose $|\mathfrak{M}|$ is enumerable, and let m_0, m_1, \dots , be an enumeration. By removing repetitions 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

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

Suppose $M \subseteq |\mathfrak{M}|$ is arbitrary. Suppose further that $\mathfrak{M}, s[M/X] \models (X(z) \wedge \forall x (X(x) \rightarrow X(u(x))))$. Then $s[M/X](z) \in M$ and whenever $x \in M$, also $(s[M/X](u))(x) \in M$. In other words, since $s[M/X] \sim_X s$, $m_0 \in M$ 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 $\mathfrak{M}, s[M/X] \models \forall x X(x)$. Since $M \subseteq |\mathfrak{M}|$ was arbitrary, we are done: $\mathfrak{M} \models \text{Count}$.

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

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

for some s . Let $m = s(z)$ and $f = s(u)$ and consider $M = \{m, f(m), f(f(m)), \dots\}$. M so defined is clearly enumerable. Then

$$\mathfrak{M}, s[M/X] \models (X(z) \wedge \forall x (X(x) \rightarrow X(u(x)))) \rightarrow \forall x X(x)$$

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

Problem syn.3. The sentence $\text{Inf} \wedge \text{Count}$ 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.

Photo Credits

Bibliography