6 The semantics of second-order logic

The languages $\mathcal{L}^2_{\mathrm{C}}$ of second order logic.

For each set C of constant symbols, we have a language $\mathcal{L}^2_{\mathrm{C}}$.

Symbols of $\mathcal{L}^2_{\mathrm{C}}$: All symbols of $\mathcal{L}^{\#}_{\mathrm{C}}$ plus n-place *predicate variables*

$$V^n_0,\ V^n_1,\ V^n_2, \ldots$$

for each $n \geq 1$. In this section, we shall speak of $v_0, v_1, \ldots$ as *individual variables*.

Remark. It is common also to have n-place function variables, but we omit them in the interest of simplicity.

Terms of $\mathcal{L}^2_{\mathrm{C}}$: The definition of terms is the same as that for $\mathcal{L}^{\#}_{\mathrm{C}}$.

Formulas of $\mathcal{L}^2_{\mathrm{C}}$: Modify the definition of formulas of $\mathcal{L}^{\#}_{\mathrm{C}}$ by changing clauses (2) and (5) as follows.

(2) For each n and i, if $t_1, \ldots, t_n$ are terms, then $P^n_i t_1 \ldots t_n$ and $V^n_i t_1 \ldots t_n$ are formulas.

(5) If A is a formula and X is an individual or predicate variable, then $\forall X A$ is a formula.

Models for $\mathcal{L}^2_{\mathrm{C}}$: Models for $\mathcal{L}^2_{\mathrm{C}}$ are the same as models for $\mathcal{L}^{\#}_{\mathrm{C}}$.

Truth and logical implication for $\mathcal{L}^2_{\mathrm{C}}$:

For each model $\mathfrak{M} = (\mathbf{D}, v, \chi)$, a *variable assignment* is a function s that assigns an element of $\mathbf{D}$ to to each individual variable and an n-place relation on $\mathbf{D}$ to each n-place predicate variable. To the definition of $\mathrm{den}^s_{\mathfrak{M}}$, we add the stipulation that $\mathrm{den}^s_{\mathfrak{M}}(V^n_i) = s(V^n_i)$ for all n and i. The definition of $v^s_{\mathfrak{M}}$ is the same as that for $\mathcal{L}^{\#}_{\mathrm{C}}$, except for two changes. First, there is an extra subclause of the atomic clause (i):

(d) $v^s_{\mathfrak{M}}(V^n_i t_1 \ldots t_n) = \mathbf{T}$ if and only if $\mathrm{den}^s_{\mathfrak{M}}(V^n_i)(\mathrm{den}^s_{\mathfrak{M}}(t_1), \ldots, \mathrm{den}^s_{\mathfrak{M}}(t_n))$ holds.

Second, clause (iv) needs to be reinterpreted so that the variable x can be of either kind.

The definition of a *free occurrence* of a variable is as before, except that it now applies to both kinds of variables. *Satisfaction* and *truth* are defined as for $\mathcal{L}^{\#}_{\mathrm{C}}$, and so are *logical implication*, *validity*, and *satisfiability*.

Consider the following sentence of $\mathcal{L}^{2}_{\mathrm{C}}$.

$$\begin{aligned}\exists V^2(&\forall v_1 \exists v_2 V^2 v_1 v_2 \\ &\wedge \forall v_1 \forall v_2 (V^2 v_1 v_2 \rightarrow \neg V^2 v_2 v_1) \\ &\wedge \forall v_1 \forall v_2 \forall v_3 ((V^2 v_1 v_2 \wedge V^2 v_2 v_3) \rightarrow V^2 v_1 v_3)).\end{aligned}$$

Call this sentence Inf. The solution to Exercise 3.4 shows that Inf can be true only in a model with an infinite domain. Conversely, Inf is true in every model with an infinite domain. This is because every infinite set can be linearly ordered in such a way that there is no greatest element. If the domain $\mathbf{D}$ of $\mathfrak{M}$ is infinite, then Inf is shown to be true in $\mathfrak{M}$ by the variable assignment that assigns such a linear ordering of $\mathbf{D}$ to V^2.

Theorem 6.1. *Compactness fails for* $\mathcal{L}^{2}_{\mathrm{C}}$.

Proof. For $n \geq 2$, let B_n be the following sentence of $\mathcal{L}^{\#}_{\mathrm{C}}$ (and so of $\mathcal{L}^{2}_{\mathrm{C}}$).

$$\exists v_1 \ldots \exists v_n (v_1 \neq v_2 \wedge \ldots \wedge v_1 \neq v_n \wedge v_2 \neq v_3 \wedge \ldots \wedge v_{n-1} \neq v_n).$$

(There is a conjunct $v_i \neq v_j$ for all i and j such that $1 \leq i < j \leq n$.) For each n, B_n is true in all models whose domain has size $\geq n$ and it is false in all models whose domain has size $< n$. Let

$$\Gamma \;=\; \{\neg \mathrm{Inf}\} \cup \{B_2, B_3, B_4, \ldots\}.$$

Clearly Γ is not satisfiable. The theorem will be proved if we can show that Γ is finitely satisfiable. Let Δ be a finite subset of Γ. Let n be the largest number such that $B_n \in \Delta$. If $\mathfrak{M}$ is any model whose domain is finite and has size $\geq n$, then, Δ is true in $\mathfrak{M}$. $\square$

Remark. Since compactness holds for $\mathcal{L}^{\#}_{\mathrm{C}}$, there can be no sentence like Inf in $\mathcal{L}^{\#}_{\mathrm{C}}$, and so also there is no sentence like $\neg$Inf in $\mathcal{L}^{\#}_{\mathrm{C}}$. Indeed, there is no set of sentences in $\mathcal{L}^{\#}_{\mathrm{C}}$ that does what $\neg$Inf does:

Exercise 6.1. Prove that there is no set Σ of sentence of $\mathcal{L}^{\#}_{\mathrm{C}}$ such that Σ is true in every model with a finite domain and false in every model with an infinite domain.

Hint. Consider the union of Σ and the set of all the B_n.

Speaking roughly, let us say that a *computable* algorithm is an one that, except for limitations of program size, computer memory, and time, could be implemented as a computer program and carried out by a computer. Call a formal language *decidable* if there is a computable algorithm that, given a formula of the language as input, will output "yes" if the formula is valid and "no" if the formula is not valid. (There is precise mathematical definition of *decidable*, which we are omitting.)

The language of sentential logic is decidable. The truth-table algorithm is computable. The language of first-order logic is not decidable (even with empty C). This was proved by Church and Turing independently. Since the language of second-order logic contains that of first-order logic, the language of second-order logic cannot be decidable.

Let us call a formal language *semi-decidable* if there is a computable algorithm that, given a formula of the language as input, will output "yes" if the formula is valid and will not say "yes" (and perhaps will not even halt) otherwise.

Decidability implies semi-decidability, so the language of sentential logic is semi-decidable. The language of first-order logic (with, say, only finitely many constants) is semi-decidable. It is not hard to see that there is a computable algorithm for listing all deductions from the empty set in the system $\mathbf{FOL}_{\mathrm{C}}$ (if C is finite). Given input A, run this listing algorithm and give output "yes" if a deduction with last line A is listed. The language of second-order logic is not semi-decidable, even with C empty. This follows from the Gödel Incompleteness Theorem.

Semi-decidable languages are essentially the same as the languages for which there exist usable sound and complete systems of deduction. Thus there is no such system of deduction for second-order logic.