Chapter 12
First-Order Logic: Undecidability and Model Theory *

The chapter surveys several important theoretical results in first-order logic. In Sect. 12.1 we prove that validity in first-order logic is undecidable, a result first proved by Alonzo Church. Validity is decidable for several classes of formulas defined by syntactic restrictions on their form (Sect. 12.2). Next, we introduce model theory (Sect. 12.3): the fact that a semantic tableau has a countable number of nodes leads to some interesting results. Finally, Sect. 12.4 contains an overview of Gödel's surprising incompleteness result.

12.1 Undecidability of First-Order Logic

We show the undecidability of validity in first-order logic by reduction from a problem whose undecidability is already known, the *halting problem*: to decide whether a Turing machine will halt if started on a blank tape (Minsky (1967, Sect. 8.3.3), Manna (1974, Sect. 1-5.2)). The proof that there is no decision procedure for validity describes an algorithm that takes an arbitrary Turing machine T and generates a formula S_T in first-order logic, such that S_T is valid if and only if T halts on an blank tape. If there were a decision procedure for validity, this construction would give us an decision procedure for the halting problem.

12.1.1 Two-Register Machines

Instead of working directly with Turing machines, we work with a simpler form of automata: two-register machines. The halting problem for two-register machines is undecidable because there is a reduction from Turing machines to two-register machines.

M. Ben-Ari, *Mathematical Logic for Computer Science*,
DOI 10.1007/978-1-4471-4129-7_12, © Springer-Verlag London 2012

Definition 12.1 A *two-register machine M* consists of two registers `x` and `y` which can store natural numbers, and a program $P = \{L_0, \ldots, L_n\}$, where L_n is the instruction `halt` and for $0 \leq i < n$, L_i is one of the instructions:

- `x = x + 1;`
- `y = y + 1;`
- `if (x == 0) goto` L_j`; else x = x - 1;`
- `if (y == 0) goto` L_j`; else y = y - 1;`

An execution sequence of M is a sequence of states $s_k = (L_i, x, y)$, where L_i is the current instruction and x, y are the contents of the registers `x` and `y`. s_{k+1} is obtained from s_k by executing L_i. The initial state is $s_0 = (L_0, m, 0)$ for some m. If for some k, $s_k = (L_n, x, y)$, the computation of M halts and M has computed $y = f(m)$. ■

Theorem 12.2 *Let T be a Turing machine that computes a function f. Then there is a two-register machine M_T that computes the function f.*

Proof Minsky (1967, Sect. 14.1), Hopcroft et al. (2006, Sect. 7.8). ■

The proof shows how the contents of the tape of a Turing machine can be encoded in an (extremely large) natural number and how the modifications to the tape can be carried out when copying the contents of one register into another. Clearly, two-register machines are even more impractical than Turing machines, but it is the theoretical result that is important.

12.1.2 Church's Theorem

Theorem 12.3 (Church) *Validity in first-order logic is undecidable.*

Proof Let M be an arbitrary two-register machine. We will construct a formula S_M such that S_M is valid iff M terminates when started in the state $(L_0, 0, 0)$. The formula is:

$$S_M = \left(p_0(a,a) \wedge \bigwedge_{i=0}^{n-1} S_i \right) \rightarrow \exists z_1 \exists z_2 \, p_n(z_1, z_2),$$

where S_i is defined by cases of the instruction L_i:

L_i	S_i
`x = x + 1;`	$\forall x \forall y (p_i(x,y) \rightarrow p_{i+1}(s(x), y))$
`y = y + 1;`	$\forall x \forall y (p_i(x,y) \rightarrow p_{i+1}(x, s(y)))$
`if (x == 0) goto Lj;` `  else x = x - 1;`	$\forall x (p_i(a,x) \rightarrow p_j(a,x)) \wedge$ $\forall x \forall y (p_i(s(x),y) \rightarrow p_{i+1}(x,y))$
`if (y == 0) then goto Lj;` `  else y = y - 1;`	$\forall x (p_i(x,a) \rightarrow p_j(x,a)) \wedge$ $\forall x \forall y (p_i(x,s(y)) \rightarrow p_{i+1}(x,y))$

The predicates are $p_0, \ldots, p_n$, one for each statement in M. The intended meaning of $p_i(x, y)$ is that the computation of M is at the label L_i and the values x, y are in the two registers. The constant a is intended to mean 0 and the function s is intended to mean the successor function $s(m) = m + 1$.

s is used both for the function symbol in the formula S_M and for states in the execution of M. The meaning will be clear from the context.

We have to prove that M halts if and only if S_M is valid.

If M Halts then S_M Is Valid

Let $s_0, \ldots, s_m$ be a computation of M that halts after m steps; we need to show that S_M is valid, that is, that it is true under any interpretation for the formula. However, we need not consider every possible interpretation. If $\mathscr{I}$ is an interpretation for S_M such that $v_{\mathscr{I}}(S_i) = F$ for some $0 \leq i \leq n-1$ or such that $v_{\mathscr{I}}(p_0(a, a)) = F$, then trivially $v_{\mathscr{I}}(S_M) = T$ since the antecedent of S_M is false. Therefore, we need only consider interpretations that satisfy the antecedent of S_M. For such interpretations, we need to show that $v_{\mathscr{I}}(\exists z_1 \exists z_2 p_n(z_1, z_2)) = T$. By induction on k, we show that $v_{\mathscr{I}}(\exists z_1 \exists z_2 p_k(z_1, z_2)) = T$.

If $k = 0$, the result is trivial since $p_0(a, a) \rightarrow \exists z_1 \exists z_2 p_0(z_1, z_2)$ is valid.

Let us assume the inductive hypothesis for $k-1$ (provided that $k > 0$) and prove $v_{\mathscr{I}}(\exists z_1 \exists z_2 p_k(z_1, z_2)) = T$. We will work through the details when L_k is `x=x+1` and leave the other cases to the reader.

By assumption the antecedent is true, in particular, its subformula S_{k-1}:

$$v_{\mathscr{I}}(\forall x \forall y(p_{k-1}(x, y) \rightarrow p_k(s(x), y))) = T,$$

and by the inductive hypothesis:

$$v_{\mathscr{I}}(\exists z_1 \exists z_2 p_{k-1}(z_1, z_2)) = T,$$

from which:

$$v_{\mathscr{I}}(\exists z_1 \exists z_2 p_k(s(z_1), z_2)) = T$$

follows by reasoning in first-order logic.

Let c_1 and c_2 be the domain elements assigned to z_1 and z_2, respectively, such that $(succ(c_1), c_2) \in P_k$, where P_k is the interpretation of p_k and $succ$ is the interpretation of s. Since $c_3 = succ(c_1)$ for some domain element c_3, the existentially quantified formula in the consequent is true:

$$v_{\mathscr{I}}(\exists z_1 \exists z_2 p_k(z_1, z_2)) = T.$$

If S_M Is Valid then M Halts

Suppose that S_M is valid and consider the interpretation:

$$\mathscr{I} = (\mathscr{N}, \{P_0, \ldots, P_n\}, \{succ\}, \{0\}),$$

where *succ* is the successor function on $\mathscr{N}$, and $(x, y) \in P_i$ iff (L_i, x, y) is reached by the register machine when started in $(L_0, 0, 0)$.

We show by induction on the length of the computation that the antecedent of S_M is true in $\mathscr{I}$. The initial state is $(L_0, 0, 0)$, so $(a, a) \in P_0$ and $v_{\mathscr{I}}(p_0(a, a)) = T$. The inductive hypothesis is that in state $s_{k-1} = (L_i, x_i, y_i)$, $(x_i, y_i) \in P_i$. The inductive step is again by cases on the type of the instruction L_i. For `x=x+1`, $s_k = (L_{i+1}, succ(x_i), y_i)$ and $(succ(x_i), y_i) \in P_{i+1}$ by the definition of P_{i+1}.

Since S_M is valid, $v_{\mathscr{I}}(\exists z_1 \exists z_2 p_n(z_1, z_2)) = T$ and $v_{\mathscr{I}}(p_n(m_1, m_2)) = T$ for some $m_1, m_2 \in \mathscr{N}$. By definition, $(m_1, m_2) \in P_n$ means that M halts and computes $m_2 = f(0)$. ■

Church's Theorem holds even if the structure of the formulas is restricted:

- The formulas contain only binary predicate symbols, one constant and one unary function symbol. This follows from the structure of S_M in the proof.
- The formulas are logic programs: a set of program clauses, a set of facts and a goal clause (Chap. 11). This follows immediately since S_M is of this form.
- The formulas are pure (Mendelson, 2009, 3.6).

Definition 12.4 A formula of first-order logic is *pure* if it contains no function symbols (including constants which are 0-ary function symbols). ■

12.2 Decidable Cases of First-Order Logic

Theorem 12.5 *There are decision procedures for the validity of pure PCNF formulas whose prefixes are of one of the forms* (*where* $m, n \geq 0$):

$$\forall x_1 \cdots \forall x_n \, \exists y_1 \cdots \exists y_m,$$
$$\forall x_1 \cdots \forall x_n \, \exists y \, \forall z_1 \cdots \forall z_m,$$
$$\forall x_1 \cdots \forall x_n \, \exists y_1 \exists y_2 \, \forall z_1 \cdots \forall z_m.$$

These classes are conveniently abbreviated $\forall^*\exists^*$, $\forall^*\exists\forall^*$, $\forall^*\exists\exists\forall^*$.

The decision procedures can be found in Dreben and Goldfarb (1979). This is the best that can be done because the addition of existential quantifiers makes validity undecidable. See Lewis (1979) for proofs of the following result.

Theorem 12.6 *There are no decision procedures for the validity of pure PCNF formulas whose prefixes are of one of the forms*:

$$\exists z\, \forall x_1 \cdots \forall x_n\, \exists y_1 \cdots \exists y_m,$$
$$\forall x_1 \cdots \forall x_n\, \exists y_1 \exists y_2 \exists y_3\, \forall z_1 \cdots \forall z_m.$$

For the first prefix, the result holds even if $n = m = 1$:

$$\exists z\, \forall x_1\, \exists y_1,$$

and for the second prefix, the result holds even if $n = 0, m = 1$:

$$\exists y_1 \exists y_2 \exists y_3\, \forall z_1.$$

Even if the matrix is restricted to contain only binary predicate symbols, there is still no decision procedure.

There are other restrictions besides those on the prefix that enable decision procedures to be given (see Dreben and Goldfarb (1979)):

Theorem 12.7 *There is a decision procedure for PCNF formulas whose matrix is of one of the forms*:

1. *All conjunctions are single literals.*
2. *All conjunctions are either single atomic formulas or consists entirely of negative literals.*
3. *All atomic formulas are monadic, that is, all predicate letters are unary.*

12.3 Finite and Infinite Models

Definition 12.8 A set of formulas U has the *finite model property* iff: U is satisfiable iff it is satisfiable in an interpretation whose domain is a finite set.

Theorem 12.9 *Let* U *be a set of pure formulas of the form*:

$$\exists x_1 \cdots \exists x_k \forall y_1 \cdots \forall y_l A(x_1, \ldots, x_k, y_1, \ldots, y_l),$$

where A *is quantifier-free. Then* U *has the finite model property.*

Proof In a tableau for U, once the δ-rules have been applied to the existential quantifiers, no more existential quantifiers remain. Thus the set of constants will be finite and the tableau will terminate once all substitutions using these constants have been made for the universal quantifiers. ■

Theorem 12.10 (Löwenheim) *If a formula is satisfiable then it is satisfiable in a countable domain.*

Proof The domain D defined in the proof of completeness is countable. ■

Löwenheim's Theorem can be generalized to countable sets of formulas $U = \{A_0, A_1, A_2, \ldots\}$. Start the tableaux with formula A_0 at the root. Whenever constructing a node at depth d, add the formula A_d into its label in addition to whatever formulas are specified by the tableau rule. If the tableau does not close, eventually, every A_i will appear on the branch, and the labels will form a Hintikka set. Hintikka's Lemma and completeness can be proved as before.

Theorem 12.11 (Löwenheim–Skolem) *If a countable set of formulas is satisfiable then it is satisfiable in a countable domain.*

Uncountable sets such as the real numbers can be described by countably many axioms (formulas). Thus formulas that describe real numbers also have a countable model in addition to the standard uncountable model! Such models are called *non-standard* models.

As in propositional logic (Theorem 3.48), compactness holds.

Theorem 12.12 (Compactness) *Let U be a countable set of formulas. If all finite subsets of U are satisfiable then so is U.*

12.4 Complete and Incomplete Theories

Definition 12.13 Let $\mathscr{T}(U)$ be a theory. $\mathscr{T}(U)$ is *complete* if and only if for every closed formula A, $U \vdash A$ or $U \vdash \neg A$. $\mathscr{T}(U)$ is *incomplete* iff it is not complete, that is, iff for some closed formula A, $U \nvdash A$ and $U \nvdash \neg A$. ■

It is important not to confuse a complete *theory* with the completeness of a *deductive system*. The latter relates the syntactic concept of proof to the semantic concept of validity: a closed formula can be proved if and only if it is valid. Completeness of a theory looks at what formulas are logical consequences of a set of formulas.

In one of the most surprising results of mathematical logic, Kurt Gödel proved that *number theory* is incomplete. Number theory, first developed by Guiseppe Peano, is a first-order logic with one constant symbol 0, one binary predicate symbol $=$, one unary function symbol s representing the successor function and two binary function symbols $+, *$. A set of axioms for number theory $\mathscr{NT}$ consists of eight axioms and one axiom scheme for induction (Mendelson, 2009, 3.1).

Theorem 12.14 (Gödel's Incompleteness Theorem) *If $\mathscr{T}(\mathscr{NT})$ is consistent then $\mathscr{T}(\mathscr{NT})$ is incomplete.*

If $\mathscr{T}(\mathscr{NT})$ were inconsistent, that is, if a theorem and its negation were both provable, then by Theorem 3.43, *every* formula would be a theorem so the theory would have be of no interest whatsoever.

The detailed proof of Gödel's theorem is tedious but not too difficult. An informal justification can be found in Smullyan (1978). Here we give a sketch of the formal

proof (Mendelson, 2009, 3.4–3.5). The idea is to define a mapping, called a *Gödel numbering*, from logical objects such as formulas and proofs to natural numbers, and then to prove the following theorem.

Theorem 12.15 *There exists a formula $A(x, y)$ in $\mathcal{NT}$ with the following property: For any numbers i, j, $A(i, j)$ is true if and only if i is the Gödel number associated with some formula $B(x)$ with one free variable x, and j is the Gödel number associated with the proof of $B(i)$. Furthermore, if $A(i, j)$ is true then a proof can be constructed for these specific integers $\vdash A(i, j)$.*

Consider now the formula $C(x) = \forall y \neg A(x, y)$ which has one free variable x, and let m be the Gödel number of this formula $C(x)$. Then $C(m) = \forall y \neg A(m, y)$ means that for no y is y the Gödel number of a proof of $C(m)$!

Theorem 12.16 (Gödel) *If $\mathcal{NT}$ is consistent then $\nvdash C(m)$ and $\nvdash \neg C(m)$.*

Proof We show that assuming either $\vdash C(m)$ or $\vdash \neg C(m)$ contradicts the consistency of $\mathcal{NT}$.

- Suppose that $\vdash C(m) = \forall y \neg A(m, y)$ and compute n, the Gödel number of this proof. Then $A(m, n)$ is true and by Theorem 12.15, $\vdash A(m, n)$. Now apply Axiom 4 of first-order logic to $C(m)$ to obtain $\vdash \neg A(m, n)$. But $\vdash A(m, n)$ and $\vdash \neg A(m, n)$ contradict the consistency of $\mathcal{NT}$.
- Suppose that $\vdash \neg C(m) = \neg \forall y \neg A(m, y) = \exists y A(m, y)$. Then for some n, $A(m, n)$ is true, where n is the Gödel number of a proof of $C(m)$, that is, $\vdash C(m)$. But we assumed $\vdash \neg C(m)$ so $\mathcal{NT}$ is inconsistent. ■

12.5 Summary

The decidability of validity for first-order logic has been investigated in detail and it is possible to precisely demarcate restricted classes of formulas which are decidable from less restricted classes that are not decidable. The Löwenheim-Skolem Theorem is surprising since it means that it is impossible to characterize uncountable structures in first-order logic. Even more surprising is Gödel's incompleteness result, since it demonstrates that there are true formulas of mathematical theories that cannot be proved in the theories themselves.

12.6 Further Reading

The two sides of the decidability question are comprehensively presented by Dreben and Goldfarb (1979) and Lewis (1979). The details of Gödel numbering can be found in (Mendelson, 2009, Chap. 3) and (Monk, 1976, Chap. 3). For an introduction to model theory see (Monk, 1976, Part 4).

12.7 Exercises

12.1 Prove that a formula is satisfiable iff it is satisfiable in an infinite model.

12.2 Prove the Löwenheim-Skolem Theorem (12.11) using the construction of semantic tableaux for infinite sets of formulas.

12.3 A closed pure formula A is *n-condensable* iff every unsatisfiable conjunction of instances of the matrix of A contains an unsatisfiable subconjunction made up of n or fewer instances.

- Let A be a PCNF formula whose matrix is a conjunction of literals. Prove that A is 2-condensable.
- Let A be a PCNF formula whose matrix is a conjunction of positive literals and disjunctions of negative literals. Prove that A is $n + 1$-condensable, where n is the maximum number of literals in a disjunction.

12.4 * Prove Church's Theorem by reducing Post's Correspondence Problem to validity in first-order logic.

References

B. Dreben and W.D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. Addison-Wesley, Reading, MA, 1979.

J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation (Third Edition)*. Addison-Wesley, 2006.

H.R. Lewis. *Unsolvable Classes of Quantificational Formulas*. Addison-Wesley, Reading, MA, 1979.

Z. Manna. *Mathematical Theory of Computation*. McGraw-Hill, New York, NY, 1974. Reprinted by Dover, 2003.

E. Mendelson. *Introduction to Mathematical Logic (Fifth Edition)*. Chapman & Hall/CRC, 2009.

M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1967.

J.D. Monk. *Mathematical Logic*. Springer, 1976.

R.M. Smullyan. *What Is the Name of This Book?—The Riddle of Dracula and Other Logical Puzzles*. Prentice-Hall, 1978.