Undecidability of First-Order Logic

Guram Bezhanishvili* Lawrence S. Moss†

Contents

1 Introduction

By a *logical system*, S, we mean a set whose elements we call *formulas*, and with a special subset that we call *theorems*. Clearly this is an extremely general definition: we have not said anything about "meaning", and there are no constraints whatsoever. But even so, it should be clear that logical systems in our current sense are studied by logicians. In particular, *first-order logic* (also called *predicate logic*) really is a logical system, as is *propositional* (or *sentential*) logic.

Let S be a logical system and let φ be a formula of it. It is desirable to know whether there is an algorithm which decides whether or not φ is a theorem of S. This *decision*

*Department of Mathematical Sciences; New Mexico State University; Las Cruces, NM 88003-8001; gbezhani@nmsu.edu.

†Department of Mathematics; Indiana University; Bloomington, IN 47405-7106; lsm@cs.indiana.edu.

problem is especially interesting for logical systems that we care about, such as first-order logic and propositional logic.

The history of this decision problem goes all the way back to Gottfried Leibniz (1646–1716) and his dream of a universal computer. But formally it was first formulated more than 200 years later by David Hilbert (1862–1943). Hilbert posed a famous set of 23 problems in 1900 at the International Congress of Mathematicians, thereby exerting great influence on twentieth century mathematics. In continuation of this, in 1928 Hilbert posed three more problems, the third of which became known as *Hilbert's Entscheidungsproblem* (German for "decision problem"). In a 1928 monograph, [14], written jointly with his student Wilhelm Ackermann (1896-1962), and which became rather influential in the development of the twentieth century logic, Hilbert characterized the problem as "the main problem of mathematical logic." This quote is taken from [15, p. 113], which is the 1950 English translation of the second edition of the monograph. When we speak of *the decision problem* in this module, we mean this particular problem. We usually say which logical system we are interested in, and the main logic of interest is first-order logic. We will see that if the decision problem for *second-order* logic had a positive solution, then it would in principle be possible to solve every mathematical problem in a purely mechanical way. Even if first-order logic were decidable, it would have had dramatic consequences.

That it was unlikely that the decision problem would have a positive solution became apparent in 1931 when Kurt Gödel (1906–1978) proved his celebrated incompleteness theorems [11][1], and opened the door for many unsolvability/undecidability results in mathematics. Indeed, a negative solution to the decision problem was given by Alonzo Church (1903–1995) in 1935–36 and independently by Alan Turing (1912-1954) in 1936–37. For Church's proof we refer to [4, 6, 5] and for Turing's proof we refer to [25]. This result has since become known as *Church's Theorem* or the *Church-Turing Theorem* (which should not be confused with *Church's Thesis*, also known as the *Church-Turing Thesis*[2]).

The proofs of Church and Turing are rather different, but both proofs were influenced by Gödel's incompleteness theorems. Church's proof uses the fact that it is undecidable whether two expressions in λ-calculus are equivalent, and then reduces this to the decision problem. On the other hand, Turing develops the celebrated *Turing machines*, shows that the *halting problem* for Turing machines is undecidable, and then reduces the halting problem to the decision problem. In either case it follows that the decision problem is undecidable.

Structure of the module: The aim of this module is to explore a variation of Turing's proof in detail. Working through it will require some work from computability theory and some work pertaining to first-order logic itself. We would like to think that interested students could read this module on their own. On the other hand, we have not provided 100% of the background that someone would need. One coming in with no experience with first-order logic, or with no experience with Turing machines or some other concrete computational model, would probably need to ask questions of a teacher. Returning to the module itself, the rest of this section presents biographical sketches of Church and Turing. Sections 2 and 3 are the background sections. They may be read in either order. These are combined in Section 4 to give the main results of our module.

[1] An English translation of Gödel's paper can be found in Davis [8, pp. 4-38].

[2] For a module on Church's Thesis see [1, pp. 253-265].

Church and Turing: The biographies of Church and Turing are as contrasting as they can be. Alonzo Church was born on June 14, 1903 in Washington DC. He graduated from Princeton in 1924 with a degree in Mathematics. He continued his graduate work at Princeton, where he received his PhD in 1927 under Oswald Veblen (1880-1960). In 1927–29 Church was a National Research fellow. During this time he visited Harvard, Götingen, and Amsterdam. Upon his return, he became an Assistant Professor at Princeton, where he remained until his retirement in 1967. He was promoted to Associate Professor in 1939, and to Full Professor in 1947. After his retirement, Church moved to UCLA and became Kent Professor of Philosophy and Professor of Mathematics. He stayed at UCLA until 1990, when he retired for the second time. In 1992 Church moved to Hudson, Ohio, where he spent the remaining three years of his life. Alonzo Church died on August 11, 1995 and was buried in Princeton Cemetery.

Church's work was of major importance to Mathematical Logic. He is best known for inventing λ-*calculus*, for his proposal to identify the effectively calculable functions with the general recursive functions, which became known as *Church's Thesis*, and for his proof that the decision problem for the first-order logic is undecidable, which became known as *Church's Theorem*. In 1936 Church also founded the Journal of Symbolic Logic, which became the premier journal in the field. He also produced a large number of remarkable PhD's. Among these were such well-known scientists as Stephen Kleene (1909–1994), Barkley Rosser (1907-1989), Leon Henkin (1921–2006), Martin Davis (born 1928), Michael Rabin (born 1931), Dana Scott (born 1932), Raymond Smullyan (born 1919), and others.

Alan Turing also received his PhD from Church in 1938. Turing was born on 23 June 1912 in London. Alan showed his extraordinary abilities already very early in his life. Turing attended King's College from 1931 to 1934. He graduated from King's College with a distinguished degree. In 1936–37, as a graduate student, he published his celebrated paper [25] in which he introduced the concept of *Turing machine*, which is considered the birth of the *Theory of Computation*. In [25] Turing also showed that the halting problem for Turing machines is undecidable, and as a corollary, he arrived at the undecidability of the decision problem for first-order logic.

In 1936 Turing went to Princeton as a visiting graduate student. He stayed at Princeton for two years and completed his PhD under Church. Upon receiving his PhD Turing returned to England, where in 1939 he attended lectures by Ludwig Wittgenstein (1889–1951) on foundations of mathematics. During the Second World War, Turing was one of the main contributors in breaking the code of the Enigma machine–the famous Nazi cipher. He was also involved in building the world's first programmable digital electronic computer. In 1945 Turing was awarded Order of the British Empire (OBE) for his wartime services. From 1945 to 1947 he worked at the National Physical Laboratory on the design of the Automatic Computing Engine. In 1948 Turing joined the department of mathematics at the University of Manchester, where in 1949 he became deputy director of the computing laboratory. During this period Turing proposed an experiment, later known as the *Turing Test*, in an attempt to decide what it means for a machine to be able to "think." The last two years of his life Turing worked in mathematical biology, where his primary interest was in pattern formation.

In 1952 Turing was convicted of gross indecency. He was a homosexual, which at that time was illegal in the United Kingdom. In order to avoid jail time, Turing agreed to undergo hormonal treatment (designed to reduce libido). This has resulted in many side effects, which

plagued the rest of his life. Alan Turing died on 8 June 1954 of cyanide poisoning. His death was ruled a suicide, however some other theories exist to this day. A detailed account of Turing's life, work, and his untimely death can be found in Hodges' biography of Alan Turing [16]. There is also an account of Turing's life in the book [26] written by Turing's mother Sara.

2 Background from Logic

The decision problem concerns both logic and computability. So our work must do the same. We present background on first-order logic in this section, the main point being the definition of *satisfiability* and examples of it.

But before we do that, we must return to the discussion of logical systems and the decision problem in our Introduction. We presented logical systems in terms of their *formulas* and *theorems*. The use of the word "theorem" suggests *proofs*. Indeed, the main logical systems are propositional logic and first-order logic, and these have proof systems. Formal proof systems are *syntactic* kinds of objects in the sense that the proofs themselves are written in a stylized, ultra-precise fashion amenable to checking by computer. When compared to a flesh-and-blood proof in a mathematics paper, a formal proof looks a bit like a simulacrum, a "fake" which nevertheless has all of the essential features of what it copies. Be that as it may, mathematical proofs can be formalized, and this is what Hilbert had in mind in his formulation of the decision problem (see [15, p. 108]):

> *The following question now arises as a fundamental problem: Is it possible to determine whether or not a given statement pertaining to a field of knowledge is a consequence of the axioms?*

The key first step in our study of the decision problem is to reformulate the matter in terms of a *semantic* notion, satisfiability. Semantic notions refer to "meaning", and the meaning of a sentence in a given logical system is always defined relative to some *model* or *interpretation* of it. We will have examples of interpretations in the rest of this section. The distinction between the syntactic notion of a proof and the semantic notion of a model is the cornerstone of modern logic. The two notions are connected by an important result, the Completeness Theorem. To state it, we need to state a definition.

Definition 2.1 A sentence φ is *valid* if it is true in all models. In contrast, φ is *satisfiable* if it is true in some model.

It is interesting to wonder which comes first, the syntactic idea of proof or the semantic idea of a model. For the logical systems of interest in this module, it is usually better to think that the semantic idea comes first. Once one has ideas related to meaning, truth, and models, it is natural to ask about formal systems which represent proofs. The point of

proofs is to help us understand the valid sentences in a logical system. Although we will not go into the details of any formal proof systems, you may find examples in practically any textbook on formal logic. A good example is a textbook by Ebbinghaus, Flum, and Thomas [10]. Proof systems for first-order logic and propositional logic come in many shapes, sizes, and colors, and the exact details will not be important to us. What *is* important is that the two fundamental notions of *validity* and *provability* coincide for first-order logic.

Theorem 2.2 (Gödel 1929) A sentence in first-order logic is provable if and only if it is valid.[3]

This fundamental theorem is often neglected in favor of Gödel's better-known Incompleteness Theorems. But it is a major result. It shows that formal proof systems can indeed perfectly capture a natural semantic notion. The best-known proof of the Completeness Theorem is by the now-celebrated *Henkin Method* which appeared in a 1949 paper by Henkin [13]. A module on Henkin's Method and the Completeness Theorem can be found on our homepage: `http://www.cs.nmsu.edu/historical-projects/`

The Completeness Theorem is connected with our module, as the exercises below show. These are short but important exercises. You should do them before reading further.

Exercise 1 Let φ be a sentence in first-order logic. Show that φ is valid if and only if $\neg\varphi$ is not satisfiable, and consequently that φ is satisfiable if and only if $\neg\varphi$ is not valid.

Exercise 2 Suppose that we have an algorithm $\mathcal{A}$ to tell whether a sentence of first-order logic is satisfiable or not. Show that we can use this to get an algorithm $\mathcal{B}$ to tell whether a sentence of first-order logic is provable or not.

Conversely, suppose that we have an algorithm $\mathcal{B}$ to tell whether a sentence of first-order logic is provable or not. Show that we can use this to get an algorithm $\mathcal{A}$ to tell whether a sentence of first-order logic is satisfiable or not.

The main result in our module is that no algorithms exist for the problems mentioned in Exercise 2 just above. The importance of the formulation in terms of the semantic notion of satisfiability is that it is easier to work with models than with formal proofs. We quote the semantic reformulation of the decision problem from [15, pp. 112–113]:

∞∞∞∞∞∞∞∞∞

> *. . . there emerges the fundamental importance of determining whether or not a given formula of the predicate calculus is universally valid . . . The problem just mentioned is called the* problem of the universal validity *of a formula. . . . If a formula* $\mathfrak{A}$ *is not universally valid in any given domain of individuals, then clearly* $\overline{\mathfrak{A}}$ *is satisfiable in that domain, and conversely . . . It is customary to refer to the two equivalent problems of* universal validity *and* satisfiability *by the common name of the* decision problem *. . .we are justified in calling it the main problem of mathematical logic.*

[3]The same result holds for propositional logic. In fact, it is a consequence of completeness of first-order logic. But it was proved earlier by Emil Post (1897-1954) using the truth-tables method. Many different proofs have been published since, including the proofs by László Kalmár (1905–1976) and Willard Quine (1908-2000). A nice historical account of the development of propositional logic can be found in Church's influential textbook [7].

We point out that what Hilbert and Ackermann call "universally valid" we simply call "valid." Also, Hilbert and Ackermann use Gothic letters for denoting formulas. Moreover, in their notation $\overline{\mathfrak{A}}$ means the negation of formula $\mathfrak{A}$.

We call a set *decidable* if its characteristic function is computable. It is our goal to show that the set FO-SAT is not decidable, where

$$\text{FO-SAT} \quad = \quad \{\varphi : \varphi \text{ is a satisfiable sentence of first-order logic}\}. \tag{1}$$

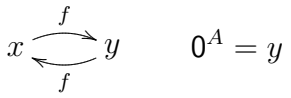
Our proof appears in Theorem 4.4.

2.1 A signature for lines

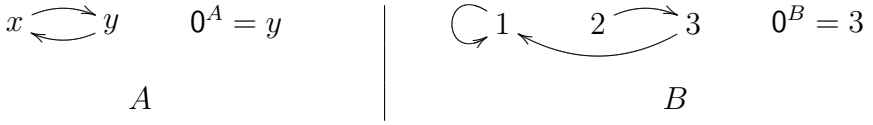
In this section by a *signature* Σ we mean a set of *function symbols*, each with a natural number called its *arity*. (Later we will want to have *relation symbols* as well.) One of the simplest signatures is the signature with a single function symbol f of arity 1, and a single constant $\mathsf{0}$. We shall see examples of Σ-*sentences* shortly, but first, let us look at some Σ-*structures*. A Σ-structure (for this particular signature Σ) is a set together with a function on it and a designated element in the set. That is, it is a triple $(X, f, 0)$, where X is any set, $f : X \to X$, and $0 \in X$. (Note that the two different fonts indicate the separation of syntax and semantics: the items in the signature are f and $\mathsf{0}$, while their interpretations in a given model are f and 0. We also will use other letters in connection with semantic interpretation, as we will see shortly.)

Although a Σ-structure technically consists of a set, a function, and a fixed element, we follow a standard practice and usually name the structure by the set. So as soon as f and 0 are specified, we usually would call the structure X instead of $(X, f, 0)$.

Here are two very simple Σ-structures, A and B: A is a two-element set $\{x, y\}$ with $f(x) = y$ and $f(y) = x$; and the interpretation of the constant is y. B is a three element set $\{1, 2, 3\}$ with $f(1) = 1$, $f(2) = 3$, and $f(3) = 1$; and the interpretation of the constant is 3. In pictures, we can draw A as



Actually, since we only have one function around, we might as well drop it from the picture and just draw A as on the left below. Similarly, then, B may be drawn as on the right.



(2)

Examples of Σ-sentences: Here is a table with some examples of Σ-sentences (again, for this very small signature Σ only). We also indicate which sentences are true in the models A and B, and which are false. In the table, $\surd$ means true and X means false.

sentence φ	A	B
$\mathsf{f}(0) = 0$	X	X
$\mathsf{f}(\mathsf{f}(0)) = \mathsf{f}(0)$	X	$\surd$
$(\exists x)\mathsf{f}(x) = x$	X	$\surd$
$(\exists y)\mathsf{f}(\mathsf{f}(y)) = x$	$\surd$	$\surd$
$(\forall z)\mathsf{f}(\mathsf{f}(z)) = z$	$\surd$	X
$(\forall x)(\exists y)\mathsf{f}(y) = x$	$\surd$	X
$(\exists x)(\forall y)x = y$	X	X

Exercise 3 As an exercise in reading first-order sentences, fill in the lines in the chart below:

sentence φ	A	B
$(\exists x)(\exists y)x \neq y$		
$(\exists x)(\exists y)(\exists z)((x \neq y) \wedge (y \neq z) \wedge (x \neq z))$		
$(\exists x)(\forall y)\mathsf{f}(\mathsf{f}(y)) = x$		
$(\exists x)(\forall y)\mathsf{f}(\mathsf{f}(x)) = y$		
$(\exists x)(\exists y)(\forall z)((\mathsf{f}(z) = x) \vee (\mathsf{f}(z) = y))$		

Justify your answers.

Isomorphic structures: Two Σ-structures $(X, f, 0_X)$ and $(Y, g, 0_Y)$ are *isomorphic* if there is a bijection $i : X \to Y$ such that for all $x \in X$, $i(f(x)) = g(i(x))$, and also $i(0_X) = 0_Y$. We write $(X, f, 0_X) \cong (Y, g, 0_Y)$ to indicate that the structures are isomorphic.

Exercise 4 Show that isomorphic models agree on all Σ-sentences. That is, if $A \cong B$, then for all φ, $A \models \varphi$ iff $B \models \varphi$.

Intuitively, the sentences cannot talk about what the elements of a structure "are," only about properties related to the function symbols. We wish to investigate the converse question: if two structures agree on the truth values of all sentences, do they have to be isomorphic?

Another way to formulate this concerns the notion of an *intended model*. Sometimes, we have a structure A in mind and we intend to describe it as fully as possible with one sentence φ. In such a situation, A would be the *intended model* of φ. By Exercise 4, the best we can do is to get a sentence φ such that $B \models \varphi$ if and only if $B \cong A$.

We now ask: is there such a sentence φ for the models A and B from (2)? Here is a preliminary result in this direction. Consider the sentence φ below:

$$\begin{array}{ll} & (\exists x)(\exists y)x \neq y \\ \wedge & \neg(\exists x)(\exists y)(\exists z)((x \neq y) \wedge (y \neq z) \wedge (x \neq z)) \\ \wedge & (\exists x)(\exists y)((x \neq y) \wedge (\mathsf{f}(x) = y) \wedge (\mathsf{f}(y) = x))) \end{array}$$

Exercise 5 Show that every model in the whole world which satisfies φ is isomorphic to the model A drawn in (2) above.

Exercise 6 Find another sentence, call it ψ, such that every model of ψ is isomorphic to the model B drawn in (2) above.

As it happens, for every finite Σ-structure A, there is a sentence φ_A such that all models of φ_A are isomorphic to A.

Exercise 7 Based on your work just above, justify this claim.

2.2 The infinite line

Now let us consider a harder case than what we have seen so far, the *infinite line* model L. It is pictured below:

$$a_0 \longrightarrow a_1 \longrightarrow \cdots \longrightarrow a_n \cdots \longrightarrow \qquad \mathsf{0}^L = a_0$$

We might take for φ_L the following sentence:

$$\begin{array}{ll} & (\forall y)(\mathsf{f}(y) \neq \mathsf{0}) \\ \wedge & (\forall z)(z \neq \mathsf{0} \rightarrow (\exists y)(\mathsf{f}(y) = z)) \\ \wedge & (\forall z)(\forall y)(\mathsf{f}(z) = \mathsf{f}(y) \rightarrow z = y) \end{array} \tag{3}$$

The first conjunct says that the distinguished point interpreting $\mathsf{0}$ is not in the image of f. The second conjunct says that every other point *is* in the image of f. (In pictures, there is exactly one point which is not the target of some arrow or other.) The third conjunct says that f is one-to-one. (In pictures, no two arrows lead to the same point.) This immediately implies that φ_L is true in L: we write this as $L \models \varphi_L$.

We might hope that every model of φ_L is isomorphic to L. However, this is not to be. Consider the model M drawn below:

$$a_0 \longrightarrow a_1 \longrightarrow \cdots \longrightarrow a_n \cdots \longrightarrow$$

$$\cdots \longrightarrow b_{-n} \cdots b_{-2} \longrightarrow b_{-1} \longrightarrow b_0 \longrightarrow b_1 \longrightarrow \cdots \longrightarrow b_n \cdots \longrightarrow$$

We set $\mathsf{0}^M = a_0$. This model M also satisfies φ_L, yet it is not isomorphic to L.

Exercise 8 Check these two assertions.

Exercise 9 (Extra Credit) Can you classify the models of φ_L? This is a relatively difficult question. Nevertheless it would be good if you give some thought to it. Can you find models similar but not isomorphic to M?

An interesting theorem of model theory called the *Löwenheim-Skolem Theorem* implies that there is no first-order sentence φ such that for all models A, $A \models \varphi$ if and only if $A \cong L$. In fact, this holds not only for L but for all infinite models whatsoever! This beautiful but a rather un-intuitive theorem was proved by Leopold Löwenheim (1878–1957) in 1915 and greatly simplified by Thoralf Skolem (1887-1963) in 1920. An even simpler proof of the theorem follows from Henkin's Method and can be found in the module on Henkin's Method on our homepage: http://www.cs.nmsu.edu/historical-projects/

2.3 Digression: stronger logical systems

There are two ways to perhaps get around the Löwenheim-Skolem Theorem. We want to mention them partly because they lead to interesting directions in logic, and partly because you should see examples of "illegal" sentences in first-order logic.

One attempt would be to use an *infinite disjunction*

$$\varphi_L \quad \wedge \quad (\forall y)((y = \mathsf{0}) \vee (y = \mathsf{f}(\mathsf{0})) \vee (y = \mathsf{f}(\mathsf{f}(\mathsf{0}))) \vee \cdots \vee (y = \mathsf{f}^n(\mathsf{0})) \vee \cdots) \tag{4}$$

The second conjunct says that there is a point x with the property that every point is $\mathsf{f}^n(x)$ for some n. This together with φ_L from (3) gives a "sentence" which does characterize L up to isomorphism. The problem is that the "sentence" above is *infinitary*. This is not allowed in first-order logic.

A second attempt would be to use *second-order logic*. This is a logical system which allows quantification over sets. So we could write

$$\varphi_L \quad \wedge \quad (\forall S)\Big((S(\mathsf{0}) \wedge (\forall x)(S(x) \to S(\mathsf{f}(x)))) \to (\forall x)S(x)\Big) \tag{5}$$

In other words, any set containing the zero element and closed under the application of the function symbol is the whole model. The specific problem in (5) is that the quantifier $(\forall S)$ is *second-order*: it quantifies over subsets of the structure rather than elements. This is not allowed in first-order logic. As with the infinite sentence in (4), the second-order sentence in (5) eliminates the "junk" in models such as M.

The use of second-order logic allows us to see why a positive solution to the general decision problem would have resulted in the solution to many problems in mathematics. For example, consider the *Twin Prime Conjecture*: there are infinitely many primes p such that $p+2$ is also prime. It is not known whether the Twin Prime Conjecture is true or false. But suppose for a second that we had a decision procedure for second-order logic. Consider the conjunction of the sentence in (5) above with the following one:

$$\begin{aligned}
& (\forall x)(\forall y)(\mathsf{p}(x,\mathsf{0}) = x \wedge \mathsf{p}(x,\mathsf{f}(y)) = \mathsf{f}(\mathsf{p}(x,y))) \\
\wedge \quad & (\forall x)(\forall y)(\mathsf{m}(x,\mathsf{0}) = \mathsf{0} \wedge \mathsf{m}(x,\mathsf{f}(y)) = \mathsf{p}(\mathsf{m}(x,y),x)) \\
\wedge \quad & (\forall x)(\exists y)\Big(x \leq y \wedge \neg(\exists w)(\exists z)((z \neq \mathsf{f}(\mathsf{0})) \wedge (w \neq \mathsf{f}(\mathsf{0})) \wedge (\mathsf{m}(z,w) = y)) \\
& \qquad\qquad \wedge \neg(\exists w)(\exists z)((z \neq \mathsf{f}(\mathsf{0})) \wedge (w \neq \mathsf{f}(\mathsf{0})) \wedge (\mathsf{m}(z,w) = \mathsf{f}(\mathsf{f}(y)))\Big)
\end{aligned} \tag{6}$$

We claim that the conjunction of (5) and (6) has a model if and only if the Twin Prime Conjecture is true. In the easy direction, if the conjecture holds, take the structure with universe N, the usual interpretation of $\mathsf{0}$, the successor function $s : N \to N$, which associates with each n its successor $n + 1$, as the interpretation of f, the addition function $+$ as the interpretation of p, and the multiplication function $\times$ as the interpretation of m. With this interpretation, (6) expresses the conjecture. So by our assumption, it is true in the model.

Exercise 10 Verify the details.

This is half of our claim. In the other direction, suppose that the conjunction of (5) and (6) has a model, say A. Then A must be isomorphic to $(N, s, 0)$, since (5) is true only of that structure, and ones isomorphic to it. So we assume that $A = N$, $\mathsf{0}^A = 0$, and f^A is the successor function. Then p and m must be the addition and multiplication functions on N, since those are the only functions that satisfy the recursion equations in (6). And then the last part of (6) tells us that the Twin Prime Conjecture is true in N.

Exercise 11 Verify the details.

This is just one example to show that *if* the decision problem were decidable, then there would be a procedure which could decide in finite time whether all sentences about numbers were true or not. And this does not stop with numbers: other structures of interest in mathematics, such as the real numbers, also have second-order characterizations up to isomorphism.

Minimal substructures: Despite the fact that our sentence φ_L in (3) does not characterize the infinite line L among the Σ-structures, we do have a partial result.

Let us take any model of φ_L, say A. Inside A, consider the set of *interpretations of all terms without variables*. Because the signature is so small, the set of terms without variables can be listed out as

$$T_0 = \{\mathsf{0}, \mathsf{f}(\mathsf{0}), \mathsf{f}(\mathsf{f}(\mathsf{0})), \ldots\}$$

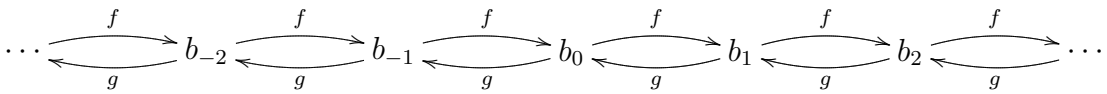
Let B be the set of elements of A which are interpretations of the terms in T_0; that is

$$B = \{\mathsf{0}^A, f(\mathsf{0}^A), f(f(\mathsf{0}^A)), \ldots\}$$

B might be called the *minimal substructure* of A. Here is why: let Σ be a first-order signature. Let C be any Σ-structure. A *substructure* of C is a subset which contains the interpretations of the constants in Σ and which is closed under the interpretations of the function symbols in Σ. It inherits the structure of a Σ-structure by interpreting relation symbols the same way as in the original structure C, and constant and function symbols similarly.

Exercise 12 Let us return to a structure A satisfying φ_L and its minimal substructure B. Show that B is naturally a Σ-structure on its own. More importantly, show that $B \cong L$.

Exercise 13 This section dealt with what we called the "infinite line," but it was a *one-way* infinite line. We used a constant $\mathsf{0}$ and a unary function symbol f. For the *two-way* infinite line $L^{\leftrightarrow}$, we could again use a constant $\mathsf{0}$ and a unary function symbol f and also add another unary function symbol, say g. We also add two relation symbols, pos, and neg. A portion of the intended structure $L^{\leftrightarrow}$ is shown below:

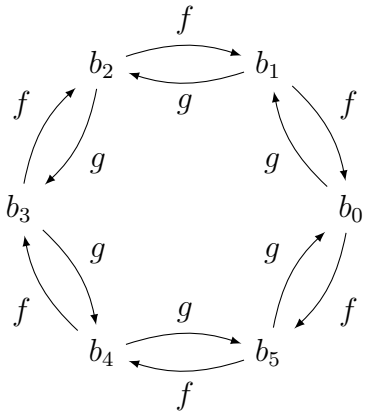


We have shown the interpretation of f as f, and the interpretation of g as g. We also interpret $\mathsf{0}$ by b_0, pos by the elements b_i with $i > 0$, and neg by the elements b_i with $i < 0$.

1. Write a single sentence χ using $\mathsf{0}$, f, g, pos, and neg whose models look as much like the intended model as possible. [You will know that your answer is correct by working the next part.]

2. Let $A \models \chi$. Check that the minimal substructure M of A is isomorphic to the two-way infinite line $L^{\leftrightarrow}$.

Note that the sentence $(\forall x)(\mathsf{f}(x) \neq \mathsf{0})$ is *not* true in $L^{\leftrightarrow}$. The point of the relation symbols pos and neg is to allow us to express more relevant facts about our structure so that we obtain the result in part (2). Here is what we mean. Suppose that we drop pos and neg and only consider $\mathsf{0}$, f, and g. Consider the structure C_6 shown below:



We may interpret $\mathsf{0}$ by any element, since the whole figure is symmetric. This structure C_6 may be generalized to C_n for all n, in the obvious way. The point is that none of the models C_n is isomorphic to our intended model $L^{\leftrightarrow}$, but they all share important properties with $L^{\leftrightarrow}$. However, any first order sentence that uses $\mathsf{0}$, f, and g which is true in $L^{\leftrightarrow}$ is true in C_n for all large enough n. (This would be a difficult fact to prove! If you know about ultraproducts, you might try computing the ultraproduct of the models C_n modulo a non-principal ultrafilter on the natural numbers.) And so the sentence χ that we are after cannot simply use the symbols $\mathsf{0}$, f, and g. However, if you use pos and neg, you will be able to write a sentence χ which has the properties we desire. We presented the models C_n to help you think about writing χ.

2.4 The upper half-plane

A natural "two-dimensional" generalization of the *two-way* line, which so to speak is a "one-dimensional" object, is the *upper half-plane* H shown in Figure 1. This structure is of central interest in connection with the decision problem for first-order logic.

The signature Σ_{uhp} contains the following symbols:

- a constant, origin;

- unary function symbols north, east, and west;

- unary relation symbols: bottom, $\mathsf{y\text{-}axis}$, $\mathsf{left\text{-}side}$, and $\mathsf{right\text{-}side}$.

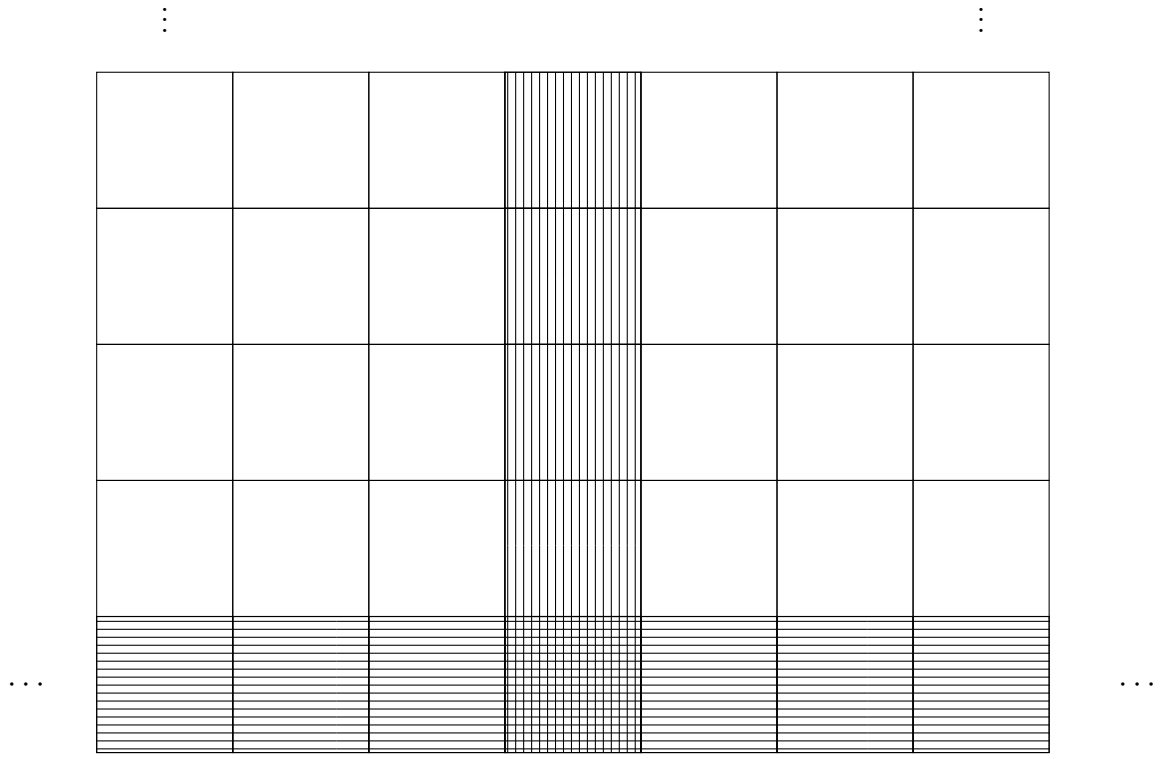


Figure 1: The upper half-plane H.

The elements of the model are the squares, not the points. The interpretation of origin is the square in the middle of the bottom line (with crosshatch shading), bottom is interpreted by the set of squares on the bottom row, the squares with the horizontal shading. The interpretation of y-axis is the set of squares with vertical lines, the squares which are above, or "north of", the origin, including the origin itself. The interpretations of the one-place functions north, east and west are again clear; they take a square and give the immediate neighbor "upwards," "rightwards," and "leftwards," respectively. The interpretation of left-side is all the squares "on the left" of the y-axis; similarly, the interpretation of right-side is all the squares "on the right" of the y-axis.

In what follows, we often use negative exponents on east and west. These mean the obvious things. For example, $\mathsf{east}^{-3}(\mathsf{origin})$ means $\mathsf{west}(\mathsf{west}(\mathsf{west}(\mathsf{origin})))$. (See Lemma 2.3 for examples of the use of this notation.)

Let χ_{uhp} be the conjunction of the following sentences:

(7) $\mathsf{bottom}(\mathsf{origin})$

(8) $(\forall x)(\mathsf{bottom}(x) \leftrightarrow ((\mathsf{origin} = x) \vee (\exists y)(\mathsf{bottom}(y) \wedge \mathsf{east}(y) = x))$

(9) $(\forall x)(\mathsf{bottom}(x) \leftrightarrow ((\mathsf{origin} = x) \vee (\exists y)(\mathsf{bottom}(y) \wedge \mathsf{west}(y) = x))$

(10) $(\forall x, y)(\mathsf{north}(x) = \mathsf{north}(y) \rightarrow x = y)$

(11) $(\forall x)\neg\mathsf{bottom}(\mathsf{north}(x))$

(12) $(\forall x)(\mathsf{north}(\mathsf{east}(x)) = \mathsf{east}(\mathsf{north}(x)))$

(13) $(\forall x)((\mathsf{west}(\mathsf{east}(x)) = x) \wedge (x = \mathsf{east}(\mathsf{west}(x))))$

(14) $(\forall x)(\mathsf{left\text{-}side}(x) \vee \mathsf{right\text{-}side}(x) \vee \mathsf{y\text{-}axis}(x))$

(15) $\mathsf{right\text{-}side}(\mathsf{east}(\mathsf{origin})) \wedge \mathsf{left\text{-}side}(\mathsf{west}(\mathsf{origin})) \wedge \neg\mathsf{right\text{-}side}(\mathsf{origin}) \wedge \neg\mathsf{left\text{-}side}(\mathsf{origin})$

(16) $(\forall x)(\mathsf{y\text{-}axis}(x) \rightarrow \neg(\mathsf{left\text{-}side}(x) \vee \mathsf{right\text{-}side}(x))$

(17) $(\forall x)(\mathsf{left\text{-}side}(x) \rightarrow \mathsf{left\text{-}side}(\mathsf{west}(x)))$

(18) $(\forall x)(\mathsf{right\text{-}side}(x) \rightarrow \mathsf{right\text{-}side}(\mathsf{east}(x)))$

Exercise 14 Explain in your own words the meaning of each conjunct of χ_{uhp}. Show that χ_{uhp} is true in H.

Exercise 15 Let $A \models \chi_{uhp}$.

1. Show that $A \models (\forall x)(\forall y)(\mathsf{east}(x) = \mathsf{east}(y) \rightarrow x = y)$, and similarly for west.

2. Show that $A \models (\forall x)(\mathsf{north}(\mathsf{west}(x)) = \mathsf{west}(\mathsf{north}(x)))$.

Lemma 2.3 Let $A \models \chi_{uhp}$, and let $a \in A$. Then the following hold in A:

1. For all $k \in \mathbb{Z}$ except 0, $\mathsf{east}^k(\mathsf{origin}) \neq \mathsf{origin}$.
2. If $\mathsf{north}(a) = \mathsf{north}^t(\mathsf{east}^k(\mathsf{origin}))$, then $t > 0$, and $a = \mathsf{north}^{t-1}(\mathsf{east}^k(\mathsf{origin}))$.
3. If $\mathsf{east}(a) = \mathsf{north}^t(\mathsf{east}^k(\mathsf{origin}))$, then $a = \mathsf{north}^t(\mathsf{east}^{k-1}(\mathsf{origin}))$.
4. If $\mathsf{west}(a) = \mathsf{north}^t(\mathsf{east}^k(\mathsf{origin}))$, then $a = \mathsf{north}^t(\mathsf{east}^{k+1}(\mathsf{origin}))$.

Exercise 16 Prove this lemma. Most of the proof is straightforward. We give a hint about how to prove the first part of (2): Suppose that $\mathsf{north}(a) = \mathsf{north}^t(\mathsf{east}^k(\mathsf{origin}))$ in A, but $t = 0$. Then $\mathsf{north}(a) = \mathsf{east}^k(\mathsf{origin})$. By (11), $\neg(\mathsf{bottom}(\mathsf{north}(a)))$; and by (7) and also (8) or (9) k times, $\mathsf{bottom}(\mathsf{east}^k(\mathsf{origin}))$. Thus we have our contradiction. Also, did your proof use all of the conjuncts in χ_{uhp}? Did you use Exercise 15?

3 Background on Turing Machines and the Halting Problem

For general background on Turing machines (TMs), we refer you to another module in the same overall project as our module, namely Section 30 on Church's Thesis in [1]. Our work will be based closely on the presentation of Turing machines by Kleene [18].

A *Turing machine* M is a tuple

$$(\mathcal{Q}, \mathcal{S}, \mathcal{R}, q_0, q_1)$$

such that

- $\mathcal{Q}$ is a finite set of *states*, $q_0 \in \mathcal{Q}$ is the *halting state*, and $q_1 \in \mathcal{Q}$ is the *starting state*.
- $\mathcal{S}$ is a finite set called the *alphabet*.
- $\mathcal{R}$ is a set of *rules* of one of the following three forms

$$(q_a, s_b) \Rightarrow (q_c, L, s_d) \qquad (q_a, s_b) \Rightarrow (q_c, C, s_d) \qquad (q_a, s_b) \Rightarrow (q_c, R, s_d).$$

- For every $q_a \in \mathcal{Q} \setminus \{q_0\}$ and every $s_b \in \mathcal{S}$ there is a unique rule of one of the three forms above with (q_a, s_b) on the left of the $\Rightarrow$ symbol.
- For every $s_b \in \mathcal{S}$ there are no rules with (q_0, s_b) on the left of the $\Rightarrow$ symbol.

We shall not enter into details of how M works on its *tape*, except to say that the state of M changes from step to step; M has a *reading head* that acts according to the current state and the last symbol under the head, and the action may be to write a symbol on the tape under the head (in place of what had been there), and also to change the state. The rules in the set $\mathcal{R}$ are the machine's instructions. For example,

$$(q_a, s_b) \Rightarrow (q_c, L, s_d)$$

would mean that if the reading head of M is in state q_a and the symbol on the current square s_b, then the symbol s_d is written on the given tape square, the head moves left one square, and the new state is q_c. We write $(q_a, s_b) \Rightarrow (q_c, C, s_d)$ for the same notion, but with the head not moving, and $(q_a, s_b) \Rightarrow (q_c, R, s_d)$ for the same notion, but with the head moving to the right one square.

One of the most important basic decision problems concerning Turing machines is whether, given a machine M, it eventually halts or not.

Theorem 3.1 (Turing 1936) *There is no TM M such that for all TMs N, if M is run with a code of N on its tape, then the following hold:*

1. *If N eventually halts when run on the empty tape, then M eventually halts with* 1 *on the tape and the reading head on the square with the* 1.
2. *If N does not eventually halt when run on the empty tape, then M eventually halts with the tape completely empty.*

We shall discuss the proof shortly, after an important point on the meaning of the theorem itself. In the meantime, we have a few other points to make. First, there is another version of Theorem 3.1, also worth stating and knowing.

Theorem 3.2 *The set* HALT *is not decidable, where*

$$\text{HALT} \quad = \quad \{M : M \textit{ is a Turing machine which halts on the empty tape}\}.$$

Coding: To make more sense of Theorems 3.1 and Theorem 3.2, you need to understand the idea of *coding* Turing machines by numbers (or by strings on some alphabet). That is, you need to understand what it would mean for a computer to take a Turing machine as input or output. If you do not know about this, you will need to look at some source such as the sections on Turing machines in [1] or [9, Sec. 4.1]. The exact details on coding are complicated at first glance, but the *idea* is fairly straightforward. We need some method of construing finite pieces of text (such as computer programs, Turing machines, or sentences in a logical language) as inputs to a computational device. One way to do it would be to put all the possible texts into alphabetical order and then just use the corresponding numbers. This is not the usual way that coding is done, but it conveys the idea. In what follows, we will not use any of the details of any coding method, just the fact that it *can* be done.

Exercise 17 Prove Theorem 3.2 from Theorem 3.1. Hint: there is a computable function f from pairs of Turing machines to Turing machines such that for all M and N, $f(M, N)$ is a machine which writes a code of N on its tape, then returns the reading head to the first symbol of the code, and finally behaves exactly as M would.

A poem which presents the ideas in Theorem 3.1: We are not going to prove Theorem 3.1 in detail. Instead, we offer a poem that explains the ideas. It is "Scooping the Loop Snooper" by Geoffrey K. Pullum [23][4]. You may find the poem at `http://ling.ed.ac.uk/~gpullum/loopsnoop.html`

Exercise 18 Read through the poem and translate it to a sound mathematical argument, using the same notation but your own language. For example, to get you started, here are the first three stanzas along with the kind of translation that you should write:

> *No general procedure for bug checks succeeds.*
> Now, I won't just assert that, I'll show where it leads:
> I will prove that although you might work till you drop,
> you cannot tell if computation will stop.
>
> For imagine we have a procedure called P
> that for specified input permits you to see
> whether specified source code, with all of its faults,

[4]The poem originally appeared in print as [22]. However we strongly encourage you to look up the revised version on author's web site.

defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behavior occurs.

Theorem *There is no program P such that for all (programs) Q and (data) R, P halts on all inputs Q and R, and has output "1" (for "true") if and only if the program Q halts on R, and output "0" (for "false") if and only if the program Q does not halt on R.*

Exercise 19 This exercise has to do with *instructions in ordinary English.* For example, if we execute the sequence of instructions

$$\textit{print "a"; print "b"; print ";"}$$

we get the output ab; If we execute

$$\textit{print the instructions to print bb}; a$$

we get *print "b"; print "b"; print ";"; print "a"*

What happens if we execute the instructions below?

print"p"; *print*"r"; *print*"i"; *print*"n"; *print*"t"; *print*"; "; *print*"t"; *print*"h"; *print*"e";
print"; "; *print*"i"; *print*"n"; *print*"s"; *print*"t"; *print*"r"; *print*"u"; *print*"c"; *print*"t";
print"i"; *print*"o"; *print*"n"; *print*"s"; *print*"; "; *print*"t"; *print*"o"; *print*"; "; *print*"p";
print"r"; *print*"i"; *print*"n"; *print*"t"; *print*"; "; *print*"w"; *print*"h"; *print*"a"; *print*"t";
print"; "; *print*"y"; *print*"o"; *print*"u"; *print*"; "; *print*"s"; *print*"e"; *print*"e"; *print*"; ";
print"b"; *print*"e"; *print*"f"; *print*"o"; *print*"r"; *print*"e"; *print*"; "; *print*"i"; *print*"t";
print the instructions to print what you see before it

Here *before it* means "in front of it." This exercise based on [20], where the idea is used to generate with computer programs that *directly output themselves*, or functions of themselves.

4 From Turing Machines to Sentences

At this point, you have read through the last two sections, giving background on first-order logic and on the undecidability of the halting problem. We now put these two sections together to prove that the decision problem for first-order logic, too, is undecidable.

The basic plan is to define a function f which takes a Turing machine M to a sentence $f(M)$ in first-order logic. The sentence $f(M)$ will be described below, in Sections 4.1–4.3. Before we do this, we want to mention the model that we are trying to capture.

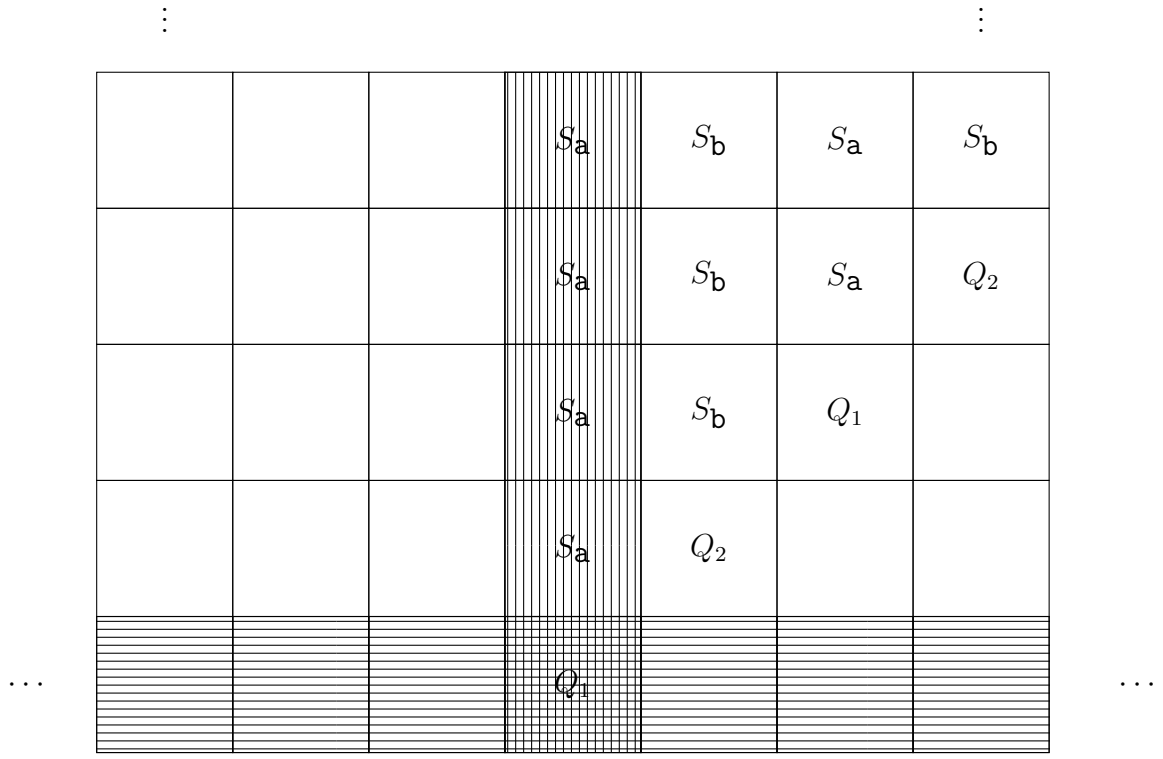


Figure 2: The intended model for the two-state Turing machine discussed in this section.

The intended models: Given a Turing machine M, we consider the two-way infinite tape at time 0, time 1, etc. Although the tape squares are not numbered, it is helpful to think of them as numbered with the integers, and with the original position of the head on square 0. We stack all of the tapes on top of each other, forever. So at this point, the model looks like the upper half-plane which we have already seen. The different alphabet symbols on the tape correspond to *one-place relation symbols*, as do the states which M uses. At each time step, the head is over exactly one of the tape squares, and the machine is in one of the states. We will take another relation symbol called head, and in each row of the upper half-plane, it will hold of exactly one square.

Example 4.1 For example, consider the TM M with two active states, q_1 and q_2. In q_1, M overprints the current square with an a, moves right and goes into q_2. In q_2, M overprints the current square with a b, moves right and goes into q_1. When started in q_1 on the empty tape, M does not halt. The model is shown in Figure 2. The interpretations of all of the symbols in χ_{uhp} are the same as those in the upper half-plane H.

Exercise 20 What are the interpretation of the symbols S_0, S_{a}, S_{b}, Q_1, and Q_2 in the model shown in Figure 2? What do these interpretations "mean"? [Hint: You might prefer to do this exercise after you read the rest of this section, or even after you encounter some

of the sentences in the next section, such as (19) and (20). These sentences are supposed to be true in models such as the one in Figure 2, and so you could get an idea about the interpretations of the various symbols by "reverse engineering" from the sentences which are supposed to be true.]

Returning to the discussion of f: At this point, we set aside our intended models. (That is, the intended model plays no formal role in the definition, but you should keep it in mind as you read on). We return to the definition of the function f. Recall that the idea is to define a function f from Turing machines to sentences. We are quite free in this definition; that is, there are many possible definitions. There are two general properties that *all* definitions of this kind must have:

1. f must be computable. To make this precise, note that we have natural functions $c : TMs \to N$ and $d : Sentences \to N$. These are both one-to-one, and their images are both *decidable*. (That is, the characteristic functions of the images are computable.) The definitions of c and d are tedious but straightforward. They begin with a coding of *finite sequences of numbers* by single numbers. For this, the prime numbers are often used, building on the Fundamental Theorem of Arithmetic. This coding is then adapted to code finite sequences from some finite or countable set, again by single numbers. The point is that all of the natural "syntactic" work with either machines or sentences must then be computable. This means that operations like telling the length of a sentence or the number of states of a TM must be computable as a function of the numerical codes.

   We are not going to enter into the details on coding in this module. Your instructor can give you as many details as you need. You can also look up detailed treatments in books such as Davis et al [9]. Most books on computability theory will indeed have a treatment of coding. We encourage you to look up as much as you need, but not to get lost in the technicalities.

   We also have a computable function $g : N \to N$ such that the diagram below commutes:

$$\begin{array}{ccc} TMs & \xrightarrow{\ c\ } & N \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ Sentences & \xrightarrow[\ d\ ]{} & N \end{array}$$

   Indeed, what it means for f to be computable is precisely that there are c, d, and g with g computable (in the usual sense), and with c and d one-to-one functions with computable image. We shall present f on an intuitive level, leaving out any discussion of c, d and g. This is the standard practice, and it gives all of the ideas.

2. M does not halt on the empty tape if and only if $f(M)$ has a model. This is probably the most important point, and the most intricate part of the argument will be devoted to establishing it.

Beyond these, we have made many choices.

1. The signature of $f(M)$ includes Σ_{uhp} from Section 2.4. This point, and the rest of the points below, will be immediate from the construction.

2. The signature of $f(M)$ also includes a relation symbol head.

3. The signature of $f(M)$ includes unary relations S_0, ..., S_j. These correspond to the symbols of M, s_0 (the blank), s_1, ..., s_j.

4. The signature of $f(M)$ includes unary relations Q_0, ..., Q_k. These correspond to the states of M, q_0 (the halting state), q_1 (the start state), ..., q_k.

4.1 Clauses for all machines

In this section and the next, we give the definition of $f(M)$. It will be a conjunction of finitely many sentences. Some of those conjuncts are independent of M and are listed in this section. In the next section, we list the conjuncts which do depend on M.

(19) Every square has a unique symbol.
$(\forall x) \bigvee_{i=0}^{j} \left(S_i(x) \wedge \bigwedge_{k \neq i} \neg S_k(x) \right)$.

(20) At time 0, all tape symbols are blank, the head is on the origin (and nowhere else), and the machine is in state Q_1 (and no other state).
$(\forall x)(\mathsf{bottom}(x) \rightarrow (S_0(x) \wedge (x = \mathsf{origin} \leftrightarrow \mathsf{head}(x)))) \wedge Q_1(\mathsf{origin}) \wedge \bigwedge_{k \neq 1} \neg Q_k(\mathsf{origin})$.

(21) If the head is not reading a given square at a given time, then the symbol on that square does not change in the next time step.

(22) The machine never halts.

Exercise 21 As you can see, (21) and (22) were written in English, leaving off the translation into logic. Provide those translations.

4.2 Clauses specific to each machine

First, suppose that the table of M contains $(s_j, q_i) \Rightarrow (s_b, L, q_d)$. (That is, when the head is on a square with s_j and the state is q_i). Then we take the sentence

(23) $(\forall x)((\mathsf{head}(x) \wedge Q_i(x) \wedge S_j(x)) \rightarrow$
$(\mathsf{head}(\mathsf{north}(\mathsf{west}(x))) \wedge Q_d(\mathsf{north}(\mathsf{west}(x))) \wedge S_b(\mathsf{north}(x))))$.

Note that the last clause really does say $S_b(\mathsf{north}(x))$ and not $S_b(\mathsf{north}(\mathsf{west}(x)))$.
Second, suppose that the table of M contains $(s_j, q_i) \Rightarrow (s_b, C, q_d)$. We take

(24) $(\forall x)((\mathsf{head}(x) \wedge Q_i(x) \wedge S_j(x)) \rightarrow (\mathsf{head}(\mathsf{north}(x)) \wedge Q_d(\mathsf{north}(x)) \wedge S_b(\mathsf{north}(x))))$.

Finally, suppose that the table of M contains $(s_j, q_i) \Rightarrow (s_b, R, q_d)$. Then we take the sentence

(25) $(\forall x)((\mathsf{head}(x) \wedge Q_i(x) \wedge S_j(x)) \rightarrow$
$(\mathsf{head}(\mathsf{north}(\mathsf{east}(x))) \wedge Q_d(\mathsf{north}(\mathsf{east}(x))) \wedge S_b(\mathsf{north}(x))))$.

We need extra clauses to say that whenever $\mathsf{head}(x)$ holds of a given x, either x is the origin, or else there was a *cause*. This clause is messy to state, but the idea of causation is clear.

(26) For all i, j except $i = 1, j = 0$,

$$
\begin{aligned}
&(\forall x)(\mathsf{head}(x) \wedge Q_i(x) \wedge S_j(x)) \rightarrow \\
&\Big[(x = \mathsf{origin}) \vee \\
&(\exists y)\Big((\mathsf{north}(\mathsf{east}(y)) = x) \wedge \textstyle\bigvee_{(Q_a,S_b)\Rightarrow(Q_i,R,S_j)} \big((\mathsf{head}(y) \wedge Q_a(y) \wedge S_b(y))\big) \\
&\quad \vee ((\mathsf{north}(y) = x) \wedge \textstyle\bigvee_{(Q_a,S_b)\Rightarrow(Q_i,C,S_j)} (\mathsf{head}(y) \wedge Q_a(y) \wedge S_b(y))) \\
&\quad \vee ((\mathsf{north}(\mathsf{west}(y)) = x) \wedge \textstyle\bigvee_{(Q_a,S_b)\Rightarrow(Q_i,L,S_j)} (\mathsf{head}(y) \wedge Q_a(y) \wedge S_b(y)))\Big)\Big]
\end{aligned}
$$

Exercise 22 This exercises has to do with clauses (23)–(26) above.

1. State these clauses in English, in your own words.
2. Recall the machine M from Example 4.1. Check that (23)–(26) are true in the model shown in Figure 2. (You will also need to recall your answer to Exercise 20 to work this exercise.)

4.3 Putting things together: the undecidability of first-order satisfiability

At long last, we are able to define the function f. Let $f(M)$ be the conjunction of χ_{uhp} (this was itself a conjunction which we saw in Section 2.4) sentences (19)–(22), and whichever of sentences (23), (24), (25), and (26) are applicable to M.

It should be clear that f is a computable function of M. This means that f is defined from a machine M not by *running* M but by *merely examining* it.

Exercise 23 Recall the machine M from Example 4.1. Give the clauses of $f(M)$ that depend on M. That is, give the sentences described in this section, specialized for this M, and written out in full glory. Hint: (24) and (23) do not apply, since the head moves right in every situation. There are two instances of (25), and one or more of (26). These are the ones to get.

Exercise 24 Explain informally how one would write a computer program which, given the coding of a Turing machine M, would output the relevant instances of (25). If you wish to enter into details, you will need to settle on some method of coding Turing machines and sentences, and for this you may adopt any reasonable method you like. But you also may work this exercise informally, appealing to intuitions about what is computable rather than the formal details.

Exercise 25 What does it mean to say that f is a computable function of M?

The main point to verify is that M does not halt on the empty tape if and only if $f(M)$ has a model.

Exercise 26 Show that if M does not halt on the empty tape, then $f(M)$ has a model. This is the easy direction. Here is a hint for how it can be done: If M does not halt, then the intended model (as described in Section 4) really is a model of $f(M)$. We use the assumption that M does not halt in order to satisfy the clause $\neg(\exists)Q_0(x)$. Although it is not really needed, this is a good place to recall our standing assumption that a non-halting M "runs forever" (see the beginning of Section 3). This makes it a little easier to build a model of $f(M)$ for such M.

For the converse, assume that $f(M)$ has a model, call it A. At first glance, there is no reason why A should be anything like the upper half-plane: A might be uncountable, it might have a lot of "junk", etc. However, since we have the upper half-plane sentences that make up χ_{uhp}, we can appeal to Lemma 2.3 for extra information.

Lemma 4.2 *Let $A \models f(M)$, and let $t \geq 0$. Then M runs for at least t steps. Also, there is a unique i such that (in M), $\mathsf{head}(\mathsf{north}^t(\mathsf{east}^i(\mathsf{origin})))$. Moreover,*

1. *There is a unique state q_a such that $Q_a(\mathsf{north}^t(\mathsf{east}^i(\mathsf{origin})))$, and $a > 0$.*
2. *At the beginning of the t^{th} step in the run of M, the state is q_a (from part (1)), the head is on square i, and for all $i \in \mathbb{Z}$, the symbol on square i is the unique s_j such that $S_j(\mathsf{north}^t(\mathsf{east}^i(\mathsf{origin})))$.*

This lemma is the heart of the matter, and so we present its proof as a series of exercises.

Exercise 27 Prove Lemma 4.2 for $t = 0$. [Hint: every TM runs for at least 0 steps. To get started, use (20) to see that $\mathsf{head}(\mathsf{origin})$, and use (20) and Lemma 2.3 to see that $\neg\mathsf{head}(\mathsf{east}^i(\mathsf{origin}))$ for $i \neq 0$. This is a start on what you need to do for $t = 0$.]

Now that the base case in Lemma 4.2 is done, assume the result for t; we prove it for $t + 1$. At the beginning of the t^{th} step, let the state be a, let the head be on square i, and let the symbol under the head be s_b. By (1) for t, $a > 0$.

Exercise 28 Say why q_a is not q_0, the halting state.

Thus there is exactly one rule of M of one of the following forms:

$$(q_a, s_b) \Rightarrow (q_c, L, s_d) \qquad (q_a, s_b) \Rightarrow (q_c, C, s_d) \qquad (q_a, s_b) \Rightarrow (q_c, R, s_d).$$

Let us assume that the rule in question is the first one, $(q_a, s_b) \Rightarrow (q_c, L, s_d)$. (Thus at the beginning of the $(t + 1)^{st}$ step, the state of M is s_d, the reading head is on square $i - 1$.)

Exercise 29 Write out the instance of (23) that applies in this situation.

Exercise 30 Use the last exercise and facts pertaining to χ_{uhp} from earlier to see that

$$\mathsf{head}(\mathsf{north}^{t+1}(\mathsf{east}^{i-1}(\mathsf{origin})))\wedge Q_d(\mathsf{head}(\mathsf{north}^{t+1}(\mathsf{east}^{i-1}(\mathsf{origin})))\wedge S_c(\mathsf{north}^{t+1}(\mathsf{east}^{i}(\mathsf{origin}))).$$

Exercise 31 We return to the induction step of Lemma 4.2. At this point, we have i as in the first desired statement, but we do not know that it is unique. Why is this so? You will need to use the uniqueness part of the induction hypothesis on t, and also clause (26) in the definition of $f(M)$.

Exercise 32 Prove the rest of the induction step.

Using Lemma 4.2 At this point, our proof of the Main Lemma is complete. One tiny point left, and then we establish the main result of our module, the undecidability of first-order satisfiability.

Lemma 4.3 *Let $A \models f(M)$. Then M does not halt.*

Exercise 33 Prove Lemma 4.3. Hint: By Lemma 4.2, for all t, M is not in the halting state at the beginning of step t.

Theorem 4.4 *The set* FO-SAT *is not decidable, where*

$$\text{FO-SAT} \quad = \quad \{\varphi : \varphi \textit{ is a satisfiable sentence of first-order logic}\}.$$

Exercise 34 Prove Theorem 4.4. Hint: Assume towards a contradiction that FO-SAT were computable. We claim that the halting set

$$\text{HALT} \quad = \quad \{M : M \text{ is a Turing machine which halts on the empty tape}\}$$

would also be computable, contrary to Theorem 3.2. The reason: if M halts, then $f(M)$ has no model (by Lemma 4.3). Conversely, if M does not halt, then as we have seen, $f(M)$ has a model. So $M \in \text{HALT}$ iff $f(M) \notin \text{FO-SAT}$. Since f is computable, we have a computable procedure to solve the halting problem: take a machine M, apply f to get a sentence, and then see if $f(M)$ is satisfiable or not.

This concludes the proof that the decision problem is not solvable. At this point, you should return to the discussion of the decision problem itself in our Introduction, to mull over the consequences of this monumental negative result.

Additional remarks and refinements: If you examine the construction of $f(M)$, you will find that the relation head is not really needed. In the intended model, each row has head true of exactly one square, and this is the same square that satisfies one of the state-relation symbols Q_a. So we simply drop head from the signature and re-write some of the clauses that go into $f(M)$. For example: (20) becomes (27), and (21) becomes (28):

(27) $Q_1(\mathsf{origin}) \wedge (\forall x)(\mathsf{bottom}(x) \wedge x \neq \mathsf{origin} \rightarrow \neg \bigvee_a Q_a(x))$.

(28) $(\forall x) \bigwedge_{i=0}^{j} (\bigvee_a Q_a(x) \wedge S_i(x) \rightarrow S_i(\mathsf{north}(x)))$.

In all the remaining sentences, head may simply be dropped. This means that our work on $f(M)$ was a little more complicated than it needed to be. But Lemma 4.2 would need to be reformulated, and we felt that the work without the head symbol might be a little more complicated than the work with that symbol.

We conclude this section with a refinement of the undecidability result. Suppose we want to restrict attention to first-order logic with relation symbols and constants only (no function symbols). We can still ask whether the decision problem is decidable. To see that it is not, note that we can replace our function symbols north, east, and west by binary relations R_n, R_w, and R_e. The meaning of $R_n(x, y)$ in the intended model is that $\mathsf{north}(x) = y$, and similarly for the other symbols.

Exercise 35 Show that $f(M)$ can be written as a sentence using only constant and relation symbols. Hint: Pay special attention to the atomic formulas with two function symbols, such as $\mathsf{north}(\mathsf{east}(x)) = y$, making sure to get their translations right. For extra credit, show that $f(M)$ can in fact be written as a sentence using only constant and relation symbols *and only three variables*.

Incidentally, it is known that the satisfiability problem for first-order logic with *only two variables*, constants, and relation symbols is decidable. See, e.g., an excellent overview of the topic in [12].

5 Notes for Instructors

The material here will take around four weeks to cover in a classroom setting. The exact amount depends on the background of the students, of course. For students with no background in logic, one would have to do more in the way of examples of sentences. For those with no background on Turing machines, it would be useful to look up Turing machine simulators on the internet. Very simple programs that the class writes could then serve as examples throughout the rest of the module, illustrating the model construction and the long sentences as well. For an on-line educational presentation of register machines (a variant of Turing machines), one might see the courseware material by the second author [20]. That presentation also allows one to present results such as the undecidability of the halting problem *without any coding whatsoever*. However, it is not based on historical sources.

We have tried hard to emphasize the ideas and the significance of the results. Surely a classroom presentation should allow for much discussion in addition to a lecture.

Our presentation avoids Gödel numbering. So students who have never seen this will have questions about what "computability" means for a function from Turing machines to sentences. We did not want to present any details, partly because different instructors have different ways to do this, and partly because these details could easily obscure the main ideas.

There are many ways to prove the undecidability of first-order satisfiability. Büchi [3] provided the first proof along the lines that we presented it. Our approach also owes a lot to work that uses the technique of *tiling* that was pioneered by Hao Wang [17, 27] and later studied by many people, including Robert Berger [2] and Raphael Robinson [24]. Indeed, one alternative approach to our work would be to first show that the tiling problem is undecidable, and then to use this as a stepping-stone to the undecidability of first-order satisfiability. See [20] for one exposition of this.

Another way would be to prove the undecidability of the *word problem* for Thue processes, or the undecidability of some other class of word problems. This is the approach of Davis et al [9] (which goes back to the work of Post [21][5] and Andrey Andreyevich Markov (1903–1979) [19]).

Yet another way to do all of our work would be to use an equational presentation of Turing machines. Here is a sketch of this approach. Given a Turing machine M with alphabet symbols $s_1, \ldots, s_j$ and states, $q_1, \ldots, q_k$, we construct a finite, purely equational set of sentences $E(M)$. The signature of these equations contains the states $q_1, \ldots, q_k$ as constants, another constant ε (representing the empty word), and finally a constant h (representing halting). Further, the alphabet symbols become *unary function symbols* $\mathsf{g}_1, \ldots, \mathsf{g}_j$ (corresponding to $s_1, \ldots, s_j$, respectively). There is one additional unary function symbol m, and one 5-place symbol f. The basic idea is that terms correspond to configurations of M with an extra "time stamp". For example, if at step 4 of a run, a machine M is in state q_a and the head is reading symbol s_b, and the tape is

$$s_c \qquad s_d \qquad s_d \qquad \underline{s_b} \qquad s_d \qquad s_c$$

(we have underlined the head position), then we would encode this by the term

$$\mathsf{f}(q_a, s_b, \mathsf{g}_d(\mathsf{g}_d(\mathsf{g}_c(\varepsilon))), \mathsf{g}_d(\mathsf{g}_c(\varepsilon)), \mathsf{m}(\mathsf{m}(\mathsf{m}(\mathsf{m}(\varepsilon)))))$$

Notice that the portion of the tape to the left of the reading head is encoded from *right-to-left*, the opposite of the standard encoding. The action of the machine is expressed in equations in this signature. For example, suppose that M has $(q_a, s_b) \Rightarrow (q_c, L, s_a)$. We then expect the tape to read

$$s_c \qquad s_d \qquad \underline{s_d} \qquad s_a \qquad s_d \qquad s_c$$

and we take the equations

$$\mathsf{f}(q_a, s_b, \mathsf{g}_d(x), y, z) \quad = \quad \mathsf{f}(q_c, s_d, x, \mathsf{g}_a(y), \mathsf{m}(z))$$

Note that x, y, and z here are *variables*; as usual, we understand equations as being universally quantified. There would be several equations corresponding to each $\Rightarrow$ assertion about

[5] It is interesting that it was Church who suggested the problem to Post!

M. One final equation would be $\mathsf{f}(q_0, s_0, x, y, z) = \mathsf{h}$; here q_0 is the halting state, and we assume that whenever M is in q_0, the symbol under the reading head is s_0. This defines a finite set $E(M)$ of equational axioms, and $M \mapsto E(M)$ is computable. The main point is to check that

$$E(M) \vdash \mathsf{f}(q_1, s_0, \varepsilon, \varepsilon, \varepsilon) = \mathsf{h}$$

if M halts when started in its starting state q_1, s_0 is under the reading head, the rest of the tape is blank, and the time counter is at 0. To prove this result takes some work. In the (more significant) left-to-right direction, it is useful to use semantic arguments based on an "intended model" . It is also useful to know that first-order provability from equational axioms coincides with provability in the smaller and more manageable *equational logic*. This use of equational logic would be a distraction for students who have not seen it, and probably the treatment of Church's Theorem in the main body of this module would be more accessible.

References

[1] J. Barnett, G. Bezhanishvili, H. Leung, J. Lodder, D. Pengelley, I. Pivkina, and D. Ranjan, *Historical projects in discrete mathematics and computer science*, Resources for Teaching Discrete Mathematics (B. Hopkins, ed.), MAA, 2008.

[2] Robert Berger, *The undecidability of the domino problem*, Mem. Amer. Math. Soc. No. **66** (1966), 72.

[3] J. Richard Büchi, *Turing-machines and the Entscheidungsproblem*, Math. Ann. **148** (1962), 201–213.

[4] Alonzo Church, *An unsolvable problem of elementary number theory, preliminary report (abstract)*, Bull. Amer. Math. Soc. **41** (1935), 332–333.

[5] ———, *A note on the Entscheidungsproblem*, J. Symbolic Logic **1** (1936), 40–41.

[6] ———, *An unsolvable problem of elementary number theory*, Amer. J. Math. **58** (1936), 345–363.

[7] ———, *Introduction to Mathematical Logic. I*, Annals of Mathematics Studies, no. 13, Princeton University Press, Princeton, N.J., 1944.

[8] Martin Davis, *The undecidable. Basic papers on undecidable propositions, unsolvable problems and computable functions*, Edited by Martin Davis, Raven Press, Hewlett, N.Y., 1965.

[9] Martin D. Davis, Ron Sigal, and Elaine J. Weyuker, *Computability, complexity, and languages: Fundamentals of theoretical computer science*, second ed., Computer Science and Scientific Computing, Academic Press, Boston, MA, 1994.

[10] H. Ebbinghaus, J. Flum, and W. Thomas, *Mathematical logic*, second ed., Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1994, Translated from the German by Margit Meßmer.

[11] Kurt Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, Monatsh. Math. Phys. **38** (1931), 173–198.

[12] Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi, *On the decision problem for two-variable first-order logic*, Bull. Symbolic Logic **3** (1997), no. 1, 53–69.

[13] Leon Henkin, *The completeness of the first-order functional calculus*, J. Symbolic Logic **14** (1949), 159–166.

[14] David Hilbert and Wilhelm Ackermann, *Grundzüge der theoretischen Logik*, Springer–Verlag, Berlin, 1928.

[15] ———, *Principles of Mathematical Logic*, Chelsea Publishing Company, New York, 1950, English translation of the second edition.

[16] Andrew Hodges, *Alan Turing: the enigma*, A Touchstone Book, Simon & Schuster, New York, 1983.

[17] A. S. Kahr, Edward F. Moore, and Hao Wang, *Entscheidungsproblem reduced to the* $\forall\exists\forall$ *case*, Proc. Nat. Acad. Sci. U.S.A. **48** (1962), 365–377.

[18] S. C. Kleene, *Introduction to metamathematics*, D. Van Nostrand Co., Inc., New York, N. Y., 1952.

[19] A. Markov, *On the impossibility of certain algorithms in the theory of associative systems*, Doklady Akad. Nauk SSSR (N.S.) **55** (1947), 583–586.

[20] Lawrence S. Moss, `1#`*: A text register machine introduction to computability theory*, courseware avaliable from `http://www.indiana.edu/~iulg/trm/`, 2007.

[21] Emil L. Post, *Recursive unsolvability of a problem of Thue*, J. Symbolic Logic **12** (1947), 1–11.

[22] Geoffrey K. Pullum, *Scooping the loop snooper*, Mathematics Magazine **73** (2000), no. 4, 319–320.

[23] ———, *Scooping the loop snooper*, paper available from `http://ling.ed.ac.uk/~gpullum/loopsnoop.html`, 2008.

[24] Raphael M. Robinson, *Undecidability and nonperiodicity for tilings of the plane*, Invent. Math. **12** (1971), 177–209.

[25] A. M. Turing, *On computable numbers, with an application to the Entscheidungsproblem*, Proc. London Math. Soc. **42** (1936), no. 2, 230–265, A correction, 43 (1937), 544–546.

[26] Sara Turing, *Alan M. Turing*, W. Heffer & Sons Ltd., Cambridge, 1959.

[27] Hao Wang, *Dominoes and the AEA case of the decision problem*, Proc. Sympos. Math. Theory of Automata (New York, 1962), Polytechnic Press of Polytechnic Inst. of Brooklyn, Brooklyn, N.Y., 1963, pp. 23–55.