Mathematical Logic

Lecturer: Mgr. Emil Jeřábek, Ph.D.
Scribe: Bc. Jindřich Novák

5 July 2024

Contents

Course remarks

Additional information, including some voluntary exercises, about this course may be found at the address

`https://users.math.cas.cz/~jerabek/teaching/mathlog.html`.

Complementary information may be found at

`https://www.karlin.mff.cuni.cz/~krajicek/mll.html`.

This course follows, or is at least consistent with, Lou van den Dries's *Mathematical Logic (Lecture Notes)* and Michael Sipser's *Introduction to the Theory of Computation* for the first and second part of the lecture as listed below respectively. The third part will not follow any particular source.

1 Syntax and semantics of logic

1.1 Propositional logic

Convention 1.1. Throughout the course, $\mathbb{N}$ denotes the set of natural numbers *including* 0. Ordered pairs and other tuples are denoted using the angle brackets $\langle x, y\rangle$.

Definition 1.2 (Strings)**.** Given an alphabet Σ, the *strings* of a given length n over Σ are elements of Σ^n. The set of all strings over Σ is

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n.$$

Notation 1.3. The length of $w \in \Sigma^*$ is denoted $|w|$; i.e., $|w|$ is the $n \in \mathbb{N}$ such that $w \in \Sigma^n$.

Given $u, v \in \Sigma^*$, their concatenation is denoted $u^\frown v$, or simply uv.

Remark 1.4. Observe $|u^\frown v| = |u| + |v|$.

Convention 1.5. Observe there is a formal distinction between a symbol $a \in \Sigma$ and a string of length one 'a' $\in \Sigma^1$. Both will, however, be denoted the same.

Definition 1.6 (Atoms, propositional formulas)**.** Let A be a set of *atoms* (or *propositional variables*). The set Prop_A of *propositional formulas* over A is the smallest subset of

$$(A \cup \{\neg, \wedge, \vee, \bot, \top, (,)\})^*$$

such that

(1) $a \in A \implies a \in \mathrm{Prop}_A$,

(2) $\phi, \psi \in \mathrm{Prop}_A \implies \neg\phi, (\phi \wedge \psi), (\phi \vee \psi), \bot, \top \in \mathrm{Prop}_A$.

Notation 1.7. We introduce the shorthands $(\phi \to \psi) = (\neg\phi \vee \psi)$, $(\phi \leftrightarrow \psi) = ((\neg\phi \vee \psi) \wedge (\neg\psi \vee \phi))$.

Lemma 1.8. *(Unique readability) For any formula $\phi \in \mathrm{Prop}_A$, exactly one of the following cases happens:*

(1) *$\phi \in A$ (ϕ is atomic).*

(2) *$\phi = \neg\phi_0$ for some $\phi_0 \in \mathrm{Prop}_A$.*

(3) *$\phi = (\phi_0 \wedge \phi_1)$ for some $\phi_0, \phi_1 \in \mathrm{Prop}_A$.*

(4) *$\phi = (\phi_0 \vee \phi_1)$ for some $\phi_0, \phi_1 \in \mathrm{Prop}_A$.*

(5) *$\phi = \bot$.*

(6) *$\phi = \top$.*

Moreover, the formulas ϕ_0 and ϕ_1 in (2)–(4) are uniquely determined by ϕ.

Remark 1.9. Note that formulas may be uniquely described by so-called *syntactic trees*. Lemma 1.8 can be generalized to show that every formula has a unique syntactic tree.

Convention 1.10. When writing formulas, we will omit outermost brackets, and, in contexts where it does not matter, brackets occurring in repeated conjunctions or disjunctions (e.g., $\phi \wedge \psi \wedge \omega$).

Notation 1.11. Given a sequence of formulas of arbitrary length $\phi_0, \ldots, \phi_{n-1}$, their repeated conjunction is abbreviated by

$$\bigwedge_{i<n} \phi_i = \phi_0 \wedge \cdots \wedge \phi_{n-1}.$$

Analogous notation is introduced for logical disjunction. For $n = 0$, this is understood as $\bigwedge_{i<0} \phi_i = \top$, $\bigvee_{i<0} \phi_i = \bot$; for $n = 1$, $\bigwedge_{i<1} \phi_i = \bigvee_{i<1} \phi_i = \phi_0$.

Lemma 1.12 (Algebraic equivalences)**.** *Conjunction and disjunction are commutative, associative, and idempotent operators up to equivalence. Moreover, $\top$ is a neutral element for conjunction, and a zero element for disjunction; dually for $\top$. We also have the lattice absorption and distributivity laws:*

$$\begin{aligned}
\phi \wedge (\psi \wedge \chi) &\equiv (\phi \wedge \psi) \wedge \chi & \phi \vee (\psi \vee \chi) &\equiv (\phi \vee \psi) \vee \chi \\
\phi \wedge \psi &\equiv \psi \wedge \phi & \phi \vee \psi &\equiv \psi \vee \phi \\
\phi \wedge \phi &\equiv \phi & \phi \vee \phi &\equiv \phi \\
\phi \wedge (\phi \vee \psi) &\equiv \phi & \phi \vee (\phi \wedge \psi) &\equiv \phi \\
\phi \wedge (\psi \vee \chi) &\equiv (\phi \wedge \psi) \vee (\phi \wedge \chi) & \phi \vee (\psi \wedge \chi) &\equiv (\phi \vee \psi) \wedge (\phi \vee \chi) \\
\phi \wedge \top &\equiv \phi & \phi \vee \bot &\equiv \phi \\
\phi \wedge \bot &\equiv \bot & \phi \vee \top &\equiv \top.
\end{aligned}$$

Lemma 1.13. *The following so called De Morgan laws hold:*

$$\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi, \qquad \neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi.$$

(We also have $\neg\neg\phi \equiv \phi$.) More generally,

$$\neg \bigvee_{i<n} \phi_i \equiv \bigwedge_{i<n} \neg\phi_i, \qquad \neg \bigwedge_{i<n} \phi_i \equiv \bigvee_{i<n} \neg\phi_i.$$

The structure $\{\mathrm{Prop}_A, \wedge, \vee, \neg, \bot, \top\}/_{\equiv}$ is a Boolean algebra.

Definition 1.14. A *propositional assignment*, or *truth assignment*, or simply an *assignment*, is a function $\alpha\colon A \to \{0,1\}$. The set of all assignments on A is denoted $\{0,1\}^A$.

Lemma 1.15 (Formula evaluation)**.** *Any truth assignment α has a unique extension $\hat{\alpha}\colon \mathrm{Prop}_A \to \{0,1\}$ such that*

(1) $\hat{\alpha}(a) = \alpha(a)$ *for any* $a \in A$,

(2) $\hat{\alpha}(\neg\phi) = 1 - \hat{\alpha}(\phi)$,

(3) $\hat{\alpha}(\phi \wedge \psi) = \min\{\hat{\alpha}(\phi), \hat{\alpha}(\psi)\}$,

(4) $\hat{\alpha}(\phi \vee \psi) = \max\{\hat{\alpha}(\phi), \hat{\alpha}(\psi)\}$,

(5) $\hat{\alpha}(\bot) = 0$ *and* $\hat{\alpha}(\top) = 1$.

Lemma 1.16. *For any sequence of formulas $\phi_0, \ldots, \phi_{n-1}$, we have*

$$\hat{\alpha}\Big(\bigvee_{i<n} \phi_i\Big) = 1 \iff \exists i < n\, \hat{\alpha}(\phi_i) = 1$$

and

$$\hat{\alpha}\Big(\bigwedge_{i<n} \phi_i\Big) = 1 \iff \forall i < n\, \hat{\alpha}(\phi_i) = 1.$$

Definition 1.17 (Satisfaction, tautology, satisfiability, equivalent formulas)**.**

(1) If $\hat{\alpha}(\phi) = 1$, then we say that α *satisfies* ϕ, or that α *is a satisfying assignment of* ϕ; this is denoted $\alpha \models \phi$.

(2) A formula ϕ is a *tautology* if every truth assignment $\alpha\colon A \to \{0,1\}$ satisfies ϕ. This fact is denoted $\models \phi$.

(3) Dually, ϕ is *satisfiable* if there exists a truth assignment $\alpha\colon A \to \{0,1\}$ satisfying ϕ.

(4) Formulas ϕ and ψ are *equivalent*, written $\phi \equiv \psi$, if

$$\forall \alpha \in \{0,1\}^A\ \hat{\alpha}(\phi) = \hat{\alpha}(\psi).$$

Observation 1.17.1. *$\phi \equiv \psi$ if and only if $\models (\phi \leftrightarrow \psi)$.*

Definition 1.18 (Satisfaction of Γ, entailment)**.** Let $\Gamma \subseteq \mathrm{Prop}_A$ be a set of propositional formulas. We say that a truth assignment α *satisfies* Γ (or that α is a model of Γ), in symbols $\alpha \models \Gamma$, if α satisfies every formula ϕ in Γ.

Γ *entails* ϕ if for each $\alpha \in \{0,1\}^A$, whenever $\alpha \models \Gamma$, then $\alpha \models \phi$. This fact is denoted $\Gamma \models \phi$.

Definition 1.19 (Boolean function)**.** A *Boolean function* is any function of the form $f\colon \{0,1\}^n \to \{0,1\}$. We can identify it with $f\colon \{0,1\}^A \to \{0,1\}$ if $|A| = n$; i.e., $A = \{a_0, \ldots, a_{n-1}\}$. A formula $\phi \in \mathrm{Prop}$ *represents* a Boolean function f if for each $\alpha \in \{0,1\}^A$ we have $f(\alpha) = \hat{\alpha}(\phi)$.

Lemma 1.20. *Every ϕ represents a unique Boolean function — the* truth-table *function $\mathrm{tt}_\phi\colon \{0,1\}^A \to \{0,1\}$ defined by $\mathrm{tt}_\phi(\alpha) = \hat{\alpha}(\phi)$.*

10 October 2023

Lemma 1.21. *If A is a finite set of atoms, every Boolean function $f\colon \{0,1\}^A \to \{0,1\}$ can be represented by a formula.*

Proof. Check that

$$f(\vec{p}) \equiv \bigvee_{\alpha \in f^{-1}(1)} \bigwedge_{i \in A} p_i^{\alpha(i)} \equiv \bigwedge_{\alpha \in f^{-1}(0)} \bigvee_{i \in A} p_i^{1-\alpha(i)}. \qquad \text{QED}$$

This is often expressed by saying that the set of connectives $\{\wedge, \vee, \neg, \bot, \top\}$ is *functionally complete* on $\{0, 1\}$.

Definition 1.22 (Literal, clause, CNF, DNF)**.**

(1) A *literal* is an atom or its negation. We write $p^1 = p$, $p^0 = \neg p$.

(2) A *clause* is a (possibly empty) disjunction of literals.

(3) A *formula in conjunctive normal form*, or *CNF*, is a conjunction of a (possibly empty) set of clauses.

(4) A *formula in disjunctive normal form*, or a *DNF*, is a (possibly empty) disjunction of a (possibly empty) conjunction of literals.

Conjunctions of literals are also called *terms*, but we will refrain from this terminology to avoid clash with Definition 1.37 below.

The proof of Lemma 1.21 actually shows:

Corollary 1.22.1. *Every Boolean function can be represented by a CNF and by a DNF.*

Corollary 1.22.2. *Every formula is equivalent to a CNF and to a DNF.*

Definition 1.23 (Formula size)**.** The *size* of a formula ϕ, denoted $|\phi|$, is the number of atoms and connectives that occur in ϕ.

Remark 1.24. It follows from the proof of Lemma 1.21 that any Boolean function in n variables can be represented by a formula of size $O(2^n n)$. We can improve this to $O(2^n)$ by an inductive construction, expressing $f(x_0, \ldots, x_n)$ as $(\neg x_n \wedge f_0(x_0, \ldots, x_{n-1})) \vee (x_n \wedge f_1(x_0, \ldots, x_{n-1}))$. One may ask if we could do better. The answer is *no*, not by much.

In fact, there are Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ such that any formula representing f has size $\Omega(2^n/\log n)$. (Conversely, every Boolean function has a formula of size $O(2^n/\log n)$, but this small improvement takes a lot of work to prove.)

This may be proved by a simple counting argument: a formula ϕ of size s is a string of length s made of $n + 5$ possible symbols, thus the number of such formulas is $\leqslant (n+5)^s$, while there are 2^{2^n} Boolean functions. If all functions can be represented by formulas of size s, then $2^{2^n} \leqslant (n+5)^s$, which implies $2^n \leqslant s \cdot \log(n+5)$, whence $s \geqslant 2^n/\log(n+5)$.

It is an open problem to construct an explicit sequence of Boolean functions that require formulas of size more than $\Omega(n^c)$ for all constants c. We can construct functions that require formulas of cubic size, but we cannot do any better.

This falls into the field of study known as *circuit complexity*, which is related to various problems in computational complexity.

Propositional logic may seem, at first sight, trivial, but it is related to many very difficult and very intensely studied areas of mathematics.

Another open problem is the question: *given a formula, how difficult is it to compute whether it is satisfiable?* One obvious way to go about this is to brute-force the solution by trying all possible assignments. This is an inefficient algorithm, however, with computational complexity on the order of 2^n.

Another possibility is to convert the given formula to a DNF and check the satisfiability thereof. Note that it is trivial to check the satisfiability of DNFs, which can be done in polynomial time. The conversion itself, however, requires exponential time to compute.

Observe that we can easily verify that ϕ is satisfiable if we are given a satisfying assignment as a witness. Problems like this, where a positive answer has an efficiently checkable witness, are said to belong to the complexity class NP. In fact, satisfiability is a "complete" problem for NP in a suitable sense. The famous question $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ in effect asks whether there exists an efficient (i.e., polynomial-time) algorithm for satisfiability testing. It is generally conjectured that this is not the case, and in fact, that every satisfiability-testing algorithm requires time $2^{\Omega(n)}$.

Exercise 1.25. There is a CNF formula of size $O(n)$ such that any equivalent DNF has size $\Omega(2^n)$.

1.2 Completeness of propositional logic

WE SHALL WORK with Hilbert-style proof systems (also known as Frege systems): this means that a proof will be simply a sequence of formulas consisting of axioms and of formulas inferred by specified rules of inference.

It is not convenient to express such proof systems in the De Morgan language of connectives. Instead we will use $\{\to, \bot\}$ (from which all the usual logical operators may be readily expressed).

We will use the axiom schemata

(1) $(\phi \to (\psi \to \chi)) \to ((\phi \to \psi) \to (\phi \to \chi))$

(2) $\phi \to (\psi \to \phi)$

(3) $((\phi \to \bot) \to \bot) \to \phi$

and the schematic rule *modus ponens* (MP): from ϕ and $\phi \to \psi$, infer ψ.

Definition 1.26 (Proofs)**.** Let $\phi \in \mathrm{Prop}_A$ and $\Gamma \subseteq \mathrm{Prop}_A$. A *proof* (or *derivation*) of ϕ from Γ is a sequence of formulas $\phi_0, \phi_1, \ldots, \phi_s$ such that $\phi_s = \phi$ and for every $i = 0, \ldots, s$ one of the following holds:

- $\phi_i \in \Gamma$.
- ϕ_i is a logical axiom (i.e., it is an instance of one of the axiom schemata).
- There are some $j, k < i$ such that ϕ_i is derived from ϕ_j, ϕ_k using *modus ponens*.

The formula ϕ is said to be *provable from* Γ, written $\Gamma \vdash \phi$, if there exists a proof of ϕ from Γ.

A proof of ϕ from the empty set $\varnothing$ is simply called a *proof of* ϕ, and ϕ is called *provable*, written $\vdash \phi$.

Our present goal is to prove the following theorem:

Theorem 1.27 (Soundness and completeness)**.** *For all* $\phi \in \mathrm{Prop}_A$, $\Gamma \subseteq \mathrm{Prop}_A$*:*

$$\Gamma \vdash \phi \iff \Gamma \models \phi.$$

The soundness theorem is the left-to-right implication, and the completeness theorem proper is the right-to-left implication. We note that soundness is proved easily and will be left as an exercise.

Lemma 1.28 (Deduction)**.**

$$\Gamma, \phi \vdash \psi \iff \Gamma \vdash \phi \to \psi.$$

Proof. The right-to-left implication is trivial (it follows by a single application of *modus ponens*). We shall now prove the left-to-right implication. By assumption, there is a proof π of ψ from $\Gamma \cup \{\phi\}$. We will show $\Gamma \vdash \phi \to \psi$ by induction on the length of π. We distinguish the ways how ψ could have been derived:

(1) Suppose ψ is either a logical axiom or $\psi \in \Gamma$; then $\Gamma \vdash \psi$, and since $\psi \to (\phi \to \psi)$ is a propositional axiom, MP yields $\Gamma \vdash \phi \to \psi$.

(2) Suppose $\psi = \phi$ (i.e., $\psi \in \{\phi\}$). The following proof shows $\Gamma \vdash \phi \to \phi$:

$(\phi \to ((\phi \to \phi) \to \phi)) \to ((\phi \to (\phi \to \phi)) \to (\phi \to \phi))$	ax. (1)
$\phi \to ((\phi \to \phi) \to \phi)$	ax. (2)
$(\phi \to (\phi \to \phi)) \to (\phi \to \phi)$	(MP)
$\phi \to (\phi \to \phi)$	ax. (2)
$\phi \to \phi$	(MP)

(3) Suppose ψ has been derived by MP from some ω and $\omega \to \psi$. By the induction hypothesis, $\Gamma \vdash \phi \to \omega$ and $\Gamma \vdash \phi \to (\omega \to \psi)$. We obtain $\Gamma \vdash \phi \to \psi$ using the propositional axiom

$$(\phi \to (\omega \to \psi)) \to ((\phi \to \omega) \to (\phi \to \psi))$$

and two applications of MP. QED

Corollary 1.28.1. $\Gamma \vdash \phi \iff \Gamma, (\phi \to \bot) \vdash \bot$.

Proof.

$\Rightarrow$ A simple application of *modus ponens*.

$\Leftarrow$ Using the deduction lemma,

$$\begin{aligned} \Gamma, \phi \to \bot \vdash \bot &\implies \Gamma \vdash (\phi \to \bot) \to \bot \\ &\implies \Gamma \vdash \phi, \end{aligned}$$

where the last line is obtained via MP from $((\phi \to \bot) \to \bot) \to \phi$, which is an axiom. QED

Definition 1.29 (Maximal consistent set)**.** $\Gamma \subseteq \mathrm{Prop}_A$ is *maximal consistent* if Γ is consistent but all $\Gamma \subsetneq \Delta \subseteq \mathrm{Prop}_A$ are inconsistent.

Lemma 1.30. *Let Γ be a maximal consistent set and $\phi \in \mathrm{Prop}_A$.*

(1) $\Gamma \vdash \phi \implies \phi \in \Gamma$.

(2) $\phi \notin \Gamma \iff \phi \to \bot \in \Gamma$.

Proof.

(1) If $\Gamma \vdash \phi$, then $\Gamma \cup \{\phi\}$ is still consistent. Then the maximality of Γ requires that ϕ be an element of Γ.

(2) $\Leftarrow$ $\{\phi, \phi \to \bot\} \vdash \bot$ and so $\phi \notin \Gamma$ by consistency of Γ.

$\Rightarrow$ $\phi \notin \Gamma$ implies by maximality of Γ that $\Gamma, \phi \vdash \bot$, which implies $\Gamma \vdash \phi \to \bot$ by the deduction lemma. It follows that $\phi \to \bot \in \Gamma$ by (1). QED

Lemma 1.31. *Every maximal consistent set Γ is satisfiable.*

Proof. Let Γ be maximal consistent and define an assignment $\alpha \colon A \to \{0, 1\}$ by

$$\alpha(p) = 1 \iff p \in \Gamma \qquad \text{for all } p \in A.$$

We will show that this equivalence holds for all formulas ϕ, not just for atoms:

$$\hat{\alpha}(\phi) = 1 \iff \phi \in \Gamma.$$

It will follow that $\alpha \models \Gamma$. We prove this by induction on the complexity of ϕ:

- Suppose ϕ is an atom. Then $\hat{\alpha}(\phi) = 1 \iff \phi \in \Gamma$ by definition.
- Suppose $\phi = \bot$. Then $\hat{\alpha}(\bot) = 0$, and $\bot \notin \Gamma$ because Γ is consistent.
- Suppose $\phi = (\psi \to \chi)$. We distinguish three cases:

(1) $\hat{\alpha}(\chi) = 1$, thus $\hat{\alpha}(\psi \to \chi) = 1$.

By the induction hypothesis, $\chi \in \Gamma$, whence $\Gamma \vdash \psi \to \chi$ using the axiom $\chi \to (\psi \to \chi)$. Finally then $\psi \to \chi \in \Gamma$ by Lemma 1.30.

(2) $\hat{\alpha}(\psi) = 0$, thus $\hat{\alpha}(\psi \to \chi) = 1$.

By the induction hypothesis, $\psi \notin \Gamma$, which implies $\Gamma, \psi \vdash \bot$ by the maximality of Γ. We derive $\Gamma, \psi \vdash (\chi \to \bot) \to \bot$ using the axiom $\bot \to ((\chi \to \bot) \to \bot)$, and $\Gamma, \psi \vdash \chi$ using $((\chi \to \bot) \to \bot) \to \chi$.

This gives us $\Gamma \vdash \psi \to \chi$ using the deduction lemma and, therefore, $\psi \to \chi \in \Gamma$ by Lemma 1.30.

(3) $\hat{\alpha}(\psi) = 1$ and $\hat{\alpha}(\chi) = 0$, thus $\hat{\alpha}(\psi \to \chi) = 0$.

By the induction hypothesis, it follows that $\psi \in \Gamma, \chi \notin \Gamma$, whence $\psi \to \chi \notin \Gamma$ (otherwise $\Gamma \supseteq \{\psi, \psi \to \chi\} \vdash \chi$ by MP, which implies $\chi \in \Gamma$, *quod non*). QED

Lemma 1.32. *Every consistent set $\Gamma \subseteq \mathrm{Prop}_A$ can be extended to a maximal consistent $\tilde{\Gamma} \subseteq \mathrm{Prop}_A$.*

Proof. We will use Zorn's Lemma. For the partial order, take all consistent extensions $\Delta \supseteq \Gamma$ ordered by inclusion. The union of any, possibly infinite, chain (= linearly ordered set) C of consistent extensions is a consistent extension. This is due to the fact that a proof contains only finitely many formulas: if $\bigcup C$ were not consistent, contradiction could be obtained from a finite subset of $\bigcup C$; since C is linearly ordered, such a finite set would be included in some $\Delta \in C$, which would contradict the consistency of Δ. Thus every chain has an upper bound. Consequently, there exists a maximal element, which is a maximal consistent extension of Γ. QED

Theorem 1.33 (Propositional completeness theorem)**.** *Let $\Gamma \subseteq \mathrm{Prop}_A$ and $\phi \in \mathrm{Prop}_A$. Then*

$$\Gamma \vdash \phi \iff \Gamma \models \phi.$$

Proof. Showing $\Gamma \vdash \phi$ implies $\Gamma \models \phi$ is left as an exercise to the reader. For the converse implication, assume $\Gamma \nvdash \phi$. Then $\Gamma \cup \{\phi \to \bot\}$ is a consistent theory by Corollary 1.28.1, and we may extend it to a maximal consistent theory $\tilde{\Gamma}$ by Lemma 1.32. There exists an assignment α satisfying $\tilde{\Gamma}$ by Lemma 1.31. Then, in particular, $\alpha \models \Gamma$, and $\hat{\alpha}(\phi \to \bot) = 1$ implies $\alpha \not\models \phi$. Consequently, $\Gamma \not\models \phi$. QED

Theorem 1.34 (Propositional compactness theorem)**.** *Let $\Gamma \subseteq \mathrm{Prop}_A$ and $\phi \in \mathrm{Prop}_A$.*

(1) *$\Gamma \models \phi$ iff there exists a finite subset $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \models \phi$.*

(2) *If all finite $\Gamma_0 \subseteq \Gamma$ are satisfiable, then Γ is satisfiable.*

Proof.

(1) $\Leftarrow$ If there exists a (finite or otherwise) subset $\Gamma_0 \models \phi$, then $\Gamma \models \phi$ trivially.

$\Rightarrow$ $\Gamma \models \phi$ implies $\Gamma \vdash \phi$ by Theorem 1.33, whence there exists a finite subset $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \phi$, as a proof of ϕ is a finite sequence of formulas. Then $\Gamma_0 \models \phi$ again by Theorem 1.33.

(2) Take $\phi = \bot$. QED

Note that the compactness theorem is a purely semantic statement and inherently does not need any underlying proof system; it may be proved by topological means only (using the topological compactness of a suitable space).

1.3 First-order logic

Definition 1.35 (Language)**.** A *language* or *signature* is a collection of relation and function symbols, each of a given arity. Formally, $L = \langle L^r, L^f, \mathrm{ar} \rangle$ where $L^r \cap L^f = \varnothing$ and $\mathrm{ar}\colon L^r \cup L^f \to \mathbb{N}$.

For any $R \in L^r$, $\mathrm{ar}(R) = n$ signifies that R is an n-ary relation symbol. Similarly, $F \in L^f$ with $\mathrm{ar}(F) = n$ is an n-ary function symbol. Nullary function symbols are called *constant symbols*. Nullary relation symbols are rarely used, but they behave essentially as propositional atoms.

Definition 1.36. An *L-structure* is $\mathcal{A} = \langle A, \{R^{\mathcal{A}} : R \in L^r\}, \{F^{\mathcal{A}} : F \in L^f\} \rangle$ where

- $A \neq \varnothing$ is the domain[1] (underlying set) of $\mathcal{A}$;
- for any $R \in L^r$ with $\mathrm{ar}(R) = n$, $R^{\mathcal{A}} \subseteq A^n$;
- for any $F \in L^f$ with $\mathrm{ar}(F) = n$, $F^{\mathcal{A}}\colon A^n \to A$.

If $s \in L^r \cup L^f$, $s^{\mathcal{A}}$ is also called the *interpretation* of s in $\mathcal{A}$.

Definition 1.37.

(1) The set of variables is $\mathrm{Var} = \{v_n : n \in \mathbb{N}\}$. We will denote variables with lowercase letters x, y, z, ...

(2) The set Term_L of L-terms is the least set such that every variable is a term and for every n-ary function symbol F and any terms $t_0, \ldots, t_{n-1}$, we have $F(t_0, \ldots, t_{n-1}) \in \mathrm{Term}_L$.

(3) For every n-ary relation symbol and all terms $t_0, \ldots, t_{n-1}$, the expression $R(t_0, \ldots, t_{n-1})$ is an atomic formula. For all terms t and s, $t = s$ is an atomic formula.

[1] Sometimes, especially in model theory, $A = \varnothing$ is admitted as well.

(4) Now we may define the set Form_L of formulas as the least set such that every atomic formula is a formula, and if ϕ and ψ are formulas, and x is a variable, then $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $\neg\phi$, $\top$, $\bot$, $\exists x\, \phi$, and $\forall x\, \phi$ are formulas.

Convention 1.38. In practice, we will not follow the above formalities when specifying languages, structures, terms, and formulas. We will typically write a language as a set of symbols, such as $L = \{+, -, \cdot, <\}$, where the nature of the symbols (relation/function, arity) either follows their conventional use, or is understood from the context. Likewise, we will write structures as tuples listing the domain and interpretations of the symbols, such as $\langle \mathbb{Z}, +, -, \cdot, < \rangle$; as also seen here, standard operations on common mathematical structures will be identified just by their symbol (we do not need to write $+^{\mathbb{Z}}$ instead of $+$ if no confusion arises). Finally, common binary symbols such as $+$ and $<$ will be written in the usual infix notation, e.g., $x + y$ and $x < y$ rather than $+(x, y)$ and $<(x, y)$.

An interpretation of a constant c in a structure $\mathcal{A}$ is formally supposed to be a function $c^{\mathcal{A}} \colon A^0 \to A$. Since $|A^0| = 1$, such a function is uniquely specified by just giving its single value; thus, we will identify $c^{\mathcal{A}}$ with an element of A.

Definition 1.39. An occurrence of a variable x in a formula ϕ is *bound* if it occurs inside a subformula that starts with $\exists x$ or $\forall x$. Such an occurrence is said to be *within the scope* of the latter bounding quantifier. Conversely, a non-bound occurrence is said to be *free*.

Variables of ϕ are said to be free if they have free occurrences in ϕ.

24 October 2023

Definition 1.40 (Closed, open formula)**.** A term or a formula is *closed* if it has no free variables; a closed formula is called a *sentence*. Conversely, a formula is called *open* (or *quantifier-free*) if it has no bound variables.

Definition 1.41 (Theory)**.** A *theory* is a set of sentences.

Definition 1.42 (Substitution)**.** Let s be a term and x a variable. Given a term t, we define $t(s/x)$ as the result of replacing every occurrence of x in t with s.

For a formula ϕ, by $\phi(s/x)$ we understand the result of replacing every free occurrence of x in ϕ with s.

More generally, if $s_0, s_1, \ldots, s_{n-1}$ are terms and $x_0, x_1, \ldots, x_{n-1}$ are distinct variables, then $t(s_0/x_0, \ldots, s_{n-1}/x_{n-1})$ is the result of simultaneously replacing every occurrence of each x_i with s_i.

The expression $\phi(s_0/x_0, \ldots, s_{n-1}/x_{n-1})$ is defined similarly.

Definition 1.43 (Term free for substitution)**.** A term s is *free for x in ϕ* (or more explicitly, *free for substitution for x in ϕ*) if no free occurrence of x in ϕ is in the scope of a quantifier of the form $\exists y$ or $\forall y$ where y occurs in s.

Notation 1.44. We write $t(x_0, \ldots, x_{n-1})$ or $\phi(x_0, \ldots, x_{n-1})$ to indicate that all variables that occur free in ϕ or t are among $x_0, x_1, \ldots, x_{n-1}$. Then we will

write $\phi(s_0, \ldots, s_{n-1})$ for $\phi(s_0/x_0, \ldots, s_{n-1}/x_{n-1})$ and likewise for terms.

Lemma 1.45 (Successive substitution)**.** *If $t(x_0, \ldots, x_{n-1})$ is free for y in the formula $\phi(x_0, \ldots, x_{n-1}, y)$ then $(\phi(t/y))(s_0/x_0, \ldots, s_{n-1}/x_{n-1})$, denoting successive substitution, is the same formula as (i.e., syntactically identical to) the formula $\phi(s_0, \ldots, s_{n-1}, t(s_0, \ldots, s_{n-1}))$.*

The proof is left as an exercise.

Definition 1.46 (Constant-symbol language extension)**.** For any L-structure $\mathcal{A}$ and $X \subseteq A$, let $L_X = L \cup \{\underline{a} : a \in X\}$, where each $\underline{a}$ is a new constant symbol distinct from all others and from all symbols of L.

Then $\mathcal{A}_X$ is an L_X-structure with domain A, $s^{\mathcal{A}_X} = s^{\mathcal{A}}$ for all $s \in L$, and $\underline{a}^{\mathcal{A}_X} = a$ for all $a \in X$.

Definition 1.47 (Evaluation)**.** If $\mathcal{A}$ is an L-structure, and t is a closed term, then we define $t^{\mathcal{A}} \in A$ by induction on the complexity of t:

- If $t = F(t_0, \ldots, t_{n-1})$ then $t^{\mathcal{A}} = F^{\mathcal{A}}\left(t_0^{\mathcal{A}}, \ldots, t_{n-1}^{\mathcal{A}}\right)$.

For any term $t(x_0, \ldots, x_{n-1})$, we define $t^{\mathcal{A}} \colon A^n \to A$ as $t^{\mathcal{A}}(a_0, \ldots, a_{n-1}) = (t(\underline{a_0}, \ldots, \underline{a_{n-1}}))^{\mathcal{A}_A}$.

Definition 1.48 (Satisfaction, model, logical consequence)**.** Let $\mathcal{A}$ be an L-structure. Given an L_A-sentence ϕ, we define $\mathcal{A} \models \phi$ by induction on the complexity of ϕ:

- If ϕ is $R(t_0, \ldots, t_{n-1})$ for some n-ary relation R and closed L_A-terms t_i, then we put $\mathcal{A} \models \phi$ iff $\left\langle t_0^{\mathcal{A}}, \ldots, t_{n-1}^{\mathcal{A}} \right\rangle \in R^{\mathcal{A}}$.
- If ϕ is $t = s$, $\mathcal{A} \models \phi \iff t^{\mathcal{A}} = s^{\mathcal{A}}$.
- We shall now define behaviour of $\models$ on logical operators and quantifiers:

$$\begin{aligned}
\mathcal{A} \models \phi_0 \wedge \phi_1 &\iff \mathcal{A} \models \phi_0 \text{ and } \mathcal{A} \models \phi_1,\\
\mathcal{A} \models \phi_0 \vee \phi_1 &\iff \mathcal{A} \models \phi_0 \text{ or } \mathcal{A} \models \phi_1,\\
\mathcal{A} \models \neg\phi &\iff \mathcal{A} \not\models \phi,\\
\mathcal{A} \models \exists x\, \phi &\iff \text{there exists } a \in A \text{ such that } \mathcal{A} \models \phi(\underline{a}/x),\\
\mathcal{A} \models \forall x\, \phi &\iff \text{for all } a \in A,\ \mathcal{A} \models \phi(\underline{a}/x).
\end{aligned}$$

Observe that in the last two clauses, $\phi(\underline{a}/x)$ is again an L_A-sentence.

For a not necessarily closed formula $\phi(x_0, \ldots, x_{n-1})$, we write $\mathcal{A} \models \phi$ if $\mathcal{A} \models \phi(\underline{a_0}, \ldots, \underline{a_{n-1}})$ for all $a_0, \ldots, a_{n-1} \in A$; we say that ϕ *holds in* $\mathcal{A}$, or $\mathcal{A}$ is a *model* of ϕ.

More generally, if $\Gamma \subseteq \mathrm{Form}_L$, we write $\mathcal{A} \models \Gamma$ if for all $\phi \in \Gamma$ we have $\mathcal{A} \models \phi$; we say that $\mathcal{A}$ is a *model* of Γ.

A formula ϕ is a *logical consequence* of Γ, or Γ *entails* ϕ, or ϕ *follows* from Γ, written $\Gamma \models \phi$, if every model $\mathcal{A}$ of Γ is also a model of ϕ.

Finally, ϕ is said to be *logically valid* (or a *first-order tautology*), written $\models \phi$, if $\varnothing \models \phi$ (i.e., ϕ is entailed by $\Gamma = \varnothing$); in other words, if $\mathcal{A} \models \phi$ for all structures $\mathcal{A}$.

Notation 1.49. We will henceforth cease underlining constant symbols unless needed for clarity.

Definition 1.50 (Formula equivalence)**.** L-formulas ϕ and ψ are *equivalent*, written $\phi \equiv \psi$, if for every L-structure $\mathcal{A}$ and every tuple $a_0, \ldots, a_{n-1} \in A$, we have

$$\mathcal{A} \models \phi(a_1, \ldots, a_{n-1}) \iff \mathcal{A} \models \psi(a_0, \ldots, a_{n-1}).$$

In other words, $\phi \equiv \psi$ iff $\models (\phi \leftrightarrow \psi)$.

Lemma 1.51. *Let Q be either the existential or the universal quantifier. Then*

$$\begin{aligned}
\neg\exists x\, \phi &\equiv \forall x\, \neg\phi, \\
\neg\forall x\, \phi &\equiv \exists x\, \neg\phi, \\
Qx\, \phi &\equiv Qy\, \phi(y/x) && \text{if } y \text{ does not occur in } \phi, \\
(\phi \wedge Qx\, \psi) &\equiv Qx\, (\phi \wedge \psi) \\
(\phi \vee Qx\, \psi) &\equiv Qx\, (\phi \vee \psi) && \Big\} \text{ if } x \text{ is not free in } \phi.
\end{aligned}$$

Definition 1.52. A formula $\phi(x_0, \ldots, x_{n-1})$ is in the *prenex normal form* if it has the form

$$Q_0 y_0 \cdots Q_{m-1} y_{m-1}\, \theta(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1})$$

where $Q_i \in \{\exists, \forall\}$, the formula θ is open, and the y_i are pairwise distinct variables, distinct from the x_js.

Lemma 1.53. *Every formula is equivalent to a formula in prenex normal form.*

Lemma 1.54. *If $\phi \equiv \phi'$ and ψ' results from a formula ψ by replacing some occurrences of ϕ as subformulas with ϕ', then $\psi \equiv \psi'$.*

1.4 First-order proof system

For the purposes of these notes, we shall consider the following list of axiom schemata and rules:

Propositional axioms and rules of inference

$$\begin{aligned}
&(\phi \to (\psi \to \chi)) \to ((\phi \to \psi) \to (\phi \to \chi)) \\
&\phi \to (\psi \to \phi) \\
&((\phi \to \bot) \to \bot) \to \phi \\
&\text{From } \phi \text{ and } \phi \to \psi \text{ infer } \psi && (\textit{modus ponens, MP})
\end{aligned}$$

Axioms of equality

$$x = x$$
$$x = y \wedge x = z \to y = z$$
$$x_0 = y_0 \wedge \cdots \wedge x_{n-1} = y_{n-1} \to (R(\vec{x}) \to R(\vec{y}))$$
$$x_0 = y_0 \wedge \cdots \wedge x_{n-1} = y_{n-1} \to F(\vec{x}) = F(\vec{y})$$

for each n-ary relation symbol R and n-ary function symbol F

Quantifier axioms and rules

Supposing t is free for x in ϕ:

$$\forall x\, \phi \to \phi(t/x)$$
$$\phi(t/x) \to \exists x\, \phi$$

Supposing x is not free in ψ:

From $\psi \to \phi$ infer $\quad \psi \to \forall x\, \phi$ (universal generalization, $\forall$Gen)
From $\phi \to \psi$ infer $\exists x\, \phi \to \psi$ (existential generalization, $\exists$Gen)

Definition 1.55 (Provability)**.** Let $\Gamma \subseteq \mathrm{Form}_L$ and $\phi \in \mathrm{Form}_L$. Then ϕ is *provable* from Γ, written $\Gamma \vdash \phi$, if there exists a sequence of formulas $\phi_0, \ldots, \phi_n$ (called a *proof* or *derivation* of ϕ from Γ) such that $\phi_n = \phi$, and for each $i \leqslant n$, one of the following holds:

- $\phi_i \in \Gamma$.
- ϕ_i is a logical axiom.
- ϕ_i is derived by a rule of inference from some of the ϕ_j, $j < i$.

If $\Gamma = \varnothing$, we just say that ϕ is *provable*, and write $\vdash \phi$.

31 October 2023

Exercise 1.56. If $\phi(p_0, \ldots, p_{n-1})$ is a propositional tautology, then

$$\vdash \phi(\psi_0, \ldots, \psi_{n-1})$$

for any first-order formulas $\psi_0, \ldots, \psi_{n-1}$. More generally, if

$$\phi_0(p_0, \ldots, p_{n-1}), \ldots, \phi_{m-1}(p_0, \ldots, p_{n-1}) \models \phi(p_0, \ldots)$$

for propositional formulas $\phi_0, \ldots, \phi_{m-1}, \phi$, then

$$\phi_0(\psi_0, \ldots, \psi_{n-1}), \ldots, \phi_{m-1}(\psi_0, \ldots, \psi_{n-1}) \vdash \phi(\psi_0, \ldots, \psi_{n-1}).$$

We will say that $\phi(\psi_0, \ldots)$ follows from $\phi_0(\psi_0, \ldots), \ldots, \phi_{m-1}(\psi_0, \ldots)$ *by propositional reasoning*.

Remark 1.57. Using $\forall$Gen and propositional reasoning, it is easy to derive a simpler version of the $\forall$Gen rule: $\phi \vdash \forall x\, \phi$.

Definition 1.58 (Universal closure)**.** For any formula $\phi(x_0, \ldots, x_{n-1})$, its *universal closure* $\phi^\forall$ is

$$\forall x_0 \ldots \forall x_{n-1}\, \phi(x_0, \ldots, x_{n-1}).$$

In other words, all freely occurring variables of ϕ are made bound using the universal quantifier.

Notation 1.59 ($\phi(\overline{x})$)**.** Sometimes, we shall also use the notation $\phi(\overline{x})$ to stand for some $\phi(x_0, \ldots, x_{n-1})$.

Exercise 1.60. Given an L-formula ϕ, the following hold:

$$\phi \vdash \phi^\forall \vdash \phi \qquad\qquad \phi \models \phi^\forall \models \phi.$$

Lemma 1.61 (Deduction)**.** *If Γ is a set of L-formulas, ψ is an L-formula, and ϕ is an L-sentence, then*

$$\Gamma, \phi \vdash \psi \iff \Gamma \vdash \phi \to \psi.$$

Proof. The right-to-left implication is a trivial application of MP. For the left-to-right implication, let $\phi_0, \ldots, \phi_n = \psi$ be a proof of ψ from $\Gamma \cup \{\phi\}$. We show $\Gamma \vdash \phi \to \phi_i$ by induction on i:

(1) Suppose that $\phi_i \in \Gamma \cup \{\phi\}$, or ϕ_i is an axiom, or ϕ_i is derived by MP. Then the proof is identical to that of Lemma 1.28.

(2) Suppose ϕ_i is derived by the existential generalization rule; i.e., $\phi_i = \exists x\, \alpha \to \beta$ is derived from $\phi_j = \alpha \to \beta$, $j < i$, where x is not free in β. By the induction hypothesis, $\Gamma \vdash \phi \to (\alpha \to \beta)$, and thus

$$\begin{aligned}
&\Gamma \vdash \phi \to (\alpha \to \beta) && \text{induction hypothesis}\\
&\quad\vdash \alpha \to (\phi \to \beta) && \text{propositional reasoning}\\
&\quad\vdash \exists x\, \alpha \to (\phi \to \beta) && \exists\text{Gen}\\
&\quad\vdash \phi \to (\exists x\, \alpha \to \beta) && \text{propositional reasoning.}
\end{aligned}$$

We can use $\exists$Gen because ϕ is a sentence, thus x is not free in $\phi \to \beta$.

(3) Finally, suppose ϕ_i is derived by universal generalization; i.e., that $\phi_i = \beta \to \forall x\, \alpha$ is derived from $\phi_j = \beta \to \alpha$. Similarly to the existential case, we can derive

$$\begin{aligned}
&\Gamma \vdash \phi \to (\beta \to \alpha) && \text{induction hypothesis}\\
&\quad\vdash \phi \wedge \beta \to \alpha && \text{propositional reasoning}\\
&\quad\vdash \phi \wedge \beta \to \forall x\, \alpha && \forall\text{Gen}\\
&\quad\vdash \phi \to (\beta \to \forall x\, \alpha) && \text{propositional reasoning.}
\end{aligned}$$

QED

Our main goal in Section 1 is to prove that the first-order proof system we have defined adequately captures logical consequence. Let us start with the easy part:

Theorem 1.62 (Soundness theorem)**.** *Let* $\Gamma \subseteq \mathrm{Form}_L$ *and* $\phi \in \mathrm{Form}_L$*. Then*

$$\Gamma \vdash \phi \implies \Gamma \models \phi.$$

Proof. We fix a proof, say $\phi_0, \ldots, \phi_n$, of ϕ from Γ, and let $\mathcal{A}$ be an L-structure such that $\mathcal{A} \models \Gamma$. We will show $\mathcal{A} \models \phi_i$ by induction on i, and hence $\mathcal{A} \models \phi$. In essence, we are proving that satisfaction is preserved under the rules of inference. As before, we consider the various ways ϕ_i could have been derived from some ϕ_j, $j < i$, and we analyse each case individually:

Derived propositionally. This case includes $\phi_i \in \Gamma$, ϕ_i being a logical axiom, and being derived by MP. We have already proven these cases in Theorem 1.33.

Derived by universal generalization. By the definition of ∀Gen, we need to verify that

$$\mathcal{A} \models \underbrace{\beta(\overline{x}) \to \alpha(\overline{x}, y)}_{\phi_j} \qquad \text{implies} \qquad \mathcal{A} \models \underbrace{\beta(\overline{x}) \to \forall y\, \alpha(\overline{x}, y)}_{\phi_i},$$

indicating explicitly the free variables. Note that y does not occur free in β by assumptions of the ∀Gen rule. Assuming

$$\mathcal{A} \models \beta(\overline{x}) \to \alpha(\overline{x}, y),$$

we will show

$$\mathcal{A} \models \beta(\overline{x}) \to \forall y\, \alpha(\overline{x}, y)$$

using the definition of satisfaction: let $\overline{a} \in A$ be such that $\mathcal{A} \models \beta(\overline{a})$; then we need to check $\mathcal{A} \models \forall y\, \alpha(\underline{a}, y)$.

Let $b \in A$: we have $\mathcal{A} \models \beta(\overline{a}) \to \alpha(\overline{a}, b)$, thus $\mathcal{A} \models \alpha(\overline{a}, b)$. This means $\mathcal{A} \models \forall y\, \alpha(\overline{a}, y)$ as b was arbitrary.

Derived by existential generalization. Suppose $\mathcal{A} \models \alpha(\overline{x}, y) \to \beta(\overline{x})$, we need to show $\mathcal{A} \models \exists y\, \alpha(\overline{x}, y) \to \beta(\overline{x})$. This can be shown by a similar argument as for ∀Gen.

Axiom of equality. This follows easily. For example, assume ϕ_i is the axiom

$$x_0 = y_0 \wedge \cdots \wedge x_{n-1} = y_{n-1} \to F(x_0, \ldots, x_{n-1}) = F(y_0, \ldots, y_{n-1}).$$

For every $\overline{a}, \overline{b} \in A$, if $\mathcal{A} \models \bigwedge_{i<n} a_i = b_i$, then $a_0 = b_0$ and … and $a_{n-1} = b_{n-1}$, thus $F^{\mathcal{A}}(\overline{a}) = F^{\mathcal{A}}(\overline{b})$, i.e., $\mathcal{A} \models F(\overline{a}) = F(\overline{b})$.

Quantifier axiom. Consider an axiom $\phi_i = \alpha(t/y) \to \exists y\, \alpha(\overline{x}, y)$. Recall that to postulate this axiom, we must assume that t is free for y in α. Indicating explicitly the free variables, ϕ is

$$\alpha(\overline{x}, t(\overline{x}, y)/y) \to \exists y\, \alpha(\overline{x}, y).$$

We need to show that this holds in any structure $\mathcal{A}$. Let $\overline{a}, b \in A$ be such that

$$\mathcal{A} \models \alpha(\overline{x}, t(\overline{x}, y)/y)(\underline{\overline{a}}/\overline{x}, \underline{b}/y).$$

By Lemma 1.45, this means

$$\mathcal{A} \models \alpha(\underline{\overline{a}}, t(\underline{\overline{a}}, \underline{b})).$$

Putting $c = t^{\mathcal{A}}(\overline{a}, b)$, we obtain $\mathcal{A} \models \alpha(\underline{\overline{a}}, \underline{c})$ (Exercise 1.63 below). It follows that $\mathcal{A} \models \exists y\, \alpha(\underline{\overline{a}}, y)$ by the definition of satisfaction.

The argument for axioms of the form $\forall y\, \alpha \to \alpha(t/y)$ is similar. QED

Exercise 1.63. If t is a closed L_A-term, and $a = t^{\mathcal{A}}$, then $\mathcal{A} \models \phi(\underline{a})$ iff $\mathcal{A} \models \phi(t)$ for any L_A-formula $\phi(x)$.

1.5 Completeness of first-order logic

We aim to prove the completeness theorem:

Theorem 1.64. *If Γ is a set of L-formulas and ϕ is an L-formula, then*

$$\Gamma \models \phi \implies \Gamma \vdash \phi.$$

Definition 1.65. Let T be an L-theory. Then T is said to be

- *consistent* if $T \nvdash \bot$;
- *complete* if for all L-sentences ϕ, we have $T \vdash \phi$ or $T \vdash \neg\phi$;
- *Henkin* if every existential statement has a witness: i.e., for every L-formula $\phi(x)$, there is a constant c (called the *Henkin constant* for ϕ) such that
$$T \vdash \exists x\, \phi(x) \to \phi(c).$$

An outline of the proof of Theorem 1.64 is as follows:

- Reduce it to showing that if T is a consistent theory, then T has a model.
- If $T \nvdash \bot$, there is a complete theory $\tilde{T} \supseteq T$, $\tilde{T} \nvdash \bot$, in the same language.
- If $T \nvdash \bot$, there is a Henkin theory $T_H \supseteq T$, $T_H \nvdash \bot$.
- If $T \nvdash \bot$ is complete and Henkin, there is a structure $\mathcal{A}$ such that $\mathcal{A} \models T$.

We shall proceed with the details. We start with the last point, which explains the motivation for defining Henkin theories.

Lemma 1.66. *If T is a complete and consistent Henkin theory, then T has a model.*

Proof. Let CT stand for the collection of all closed L-terms. We define an equivalence relation on CT by $t \sim s$ iff $T \vdash t = s$. The axioms of equality ensure that $\sim$ is an equivalence relation, hence we may define the quotient set $A = CT/_\sim$. If t is a closed term, let $[t]$ denote the equivalence class of t.

We define an L-structure $\mathcal{A}$ with underlying set A by

$$F^{\mathcal{A}}([t_0], \ldots, [t_{n-1}]) = [F(t_0, \ldots, t_{n-1})],$$
$$\langle [t_0], \ldots, [t_{n-1}] \rangle \in R^{\mathcal{A}} \iff T \vdash R(t_0 \ldots t_{n-1})$$

for each n-ary function symbol F, and n-ary relation symbol R. In order to make sure that $F^{\mathcal{A}}$ and $R^{\mathcal{A}}$ are well-defined, we need to check that the definitions are independent of the choice of representatives of the equivalence classes: i.e., if $[t_0] = [s_0], \ldots, [t_{n-1}] = [s_{n-1}]$, then $[F(\bar{t})] = [F(\bar{s})]$, and $T \vdash R(\bar{t}) \iff T \vdash R(\bar{s})$. This follows from the equality axioms.

We can show $t^{\mathcal{A}} = [t]$ for each $t \in CT$ by induction on the complexity of t.

We claim that

$$\mathcal{A} \models \phi \iff T \vdash \phi$$

for all sentences ϕ, which implies $\mathcal{A} \models T$. We proceed by induction on the complexity of ϕ:

Atomic formula. Suppose ϕ is $R(t_0, \ldots, t_{n-1})$. Then

$$\begin{aligned} \mathcal{A} \models R(t_0 \cdots t_{n-1}) &\iff \left\langle t_0^{\mathcal{A}}, \ldots, t_{n-1}^{\mathcal{A}} \right\rangle \in R^{\mathcal{A}} \\ &\iff \langle [t_0], \ldots, [t_{n-1}] \rangle \in R^{\mathcal{A}} \\ &\iff T \vdash R(t_0, \ldots, t_{n-1}). \end{aligned}$$

The same argument also applies with $=$ in place of R.

Negation. Suppose ϕ is $\neg\psi$. Then

$$\begin{aligned} \mathcal{A} \models \neg\psi &\iff \mathcal{A} \not\models \psi \\ &\iff T \not\vdash \psi && \text{induction hypothesis} \\ &\iff T \vdash \neg\psi && \text{completeness and consistency.} \end{aligned}$$

Conjunction. Suppose ϕ is $\phi_0 \wedge \phi_1$. Then

$$\begin{aligned} \mathcal{A} \models \phi_0 \wedge \phi_1 &\iff \mathcal{A} \models \phi_0 \text{ and } \mathcal{A} \models \phi_1 \\ &\iff T \vdash \phi_0 \text{ and } T \vdash \phi_1 && \text{induction hypothesis} \\ &\iff T \vdash \phi_0 \wedge \phi_1 && \text{propositional reasoning.} \end{aligned}$$

Disjunction. Suppose ϕ is $\phi_0 \vee \phi_1$. Then

$$\begin{aligned} \mathcal{A} \models \phi_0 \vee \phi_1 &\iff T \vdash \phi_0 \text{ or } T \vdash \phi_1 \\ &\iff T \vdash \phi_0 \vee \phi_1. \end{aligned}$$

The last equivalence follows from the completeness of T: if $T \vdash \phi_0 \vee \phi_1$ and $T \nvdash \phi_0$, then $T \vdash \neg\phi_0$ by completeness, hence $T \vdash \phi_1$ by propositional reasoning.

Existential quantification. Suppose ϕ is $\exists x\, \psi(x)$. Then

$$\begin{aligned}
\mathcal{A} \models \exists x\, \psi(x) &\iff (\exists t \in CT)\, \mathcal{A} \models \psi(\underline{[t]}) \\
&\iff (\exists t \in CT)\, \mathcal{A} \models \psi(t) && \text{using } t^{\mathcal{A}} = [t] = \underline{[t]}^{\mathcal{A}} \\
&\iff (\exists t \in CT)\, T \vdash \psi(t) && \text{induction hypothesis} \\
&\iff T \vdash \exists x\, \psi(x).
\end{aligned}$$

A more thorough explanation of the last equivalence is in order. For '$\Rightarrow$', use the axiom $\psi(t) \to \exists x\, \psi(x)$; for '$\Leftarrow$', T is Henkin whence there exists a constant c such that $T \vdash \exists x\, \psi(x) \to \psi(c)$.

Universal quantification. Suppose ϕ is $\forall x\, \psi(x)$. We compute

$$\begin{aligned}
\mathcal{A} \models \forall x\, \psi(x) &\iff (\forall t \in CT)\, \mathcal{A} \models \psi(\underline{[t]}) \\
&\iff (\forall t \in CT)\, \mathcal{A} \models \psi(t) \\
&\iff (\forall t \in CT)\, T \vdash \psi(t) \\
&\iff T \vdash \forall x\, \psi(x)
\end{aligned}$$

similarly to the existential case. For the last equivalence, we need that even though the definition of Henkin theories only provides Henkin constants for existential sentences, we also obtain suitable Henkin constants for universal sentences: see Lemma 1.67 below. QED

Lemma 1.67. *If T is a Henkin L-theory, then for every L-formula $\phi(x)$, there is a constant c such that $T \vdash \phi(c) \to \forall x\, \phi(x)$.*

Proof. By assumption, there is a constant c such that $T \vdash \exists x\, \neg\phi(x) \to \neg\phi(c)$. Then we have

$T \vdash \neg\phi(x) \to \exists x\, \neg\phi(x)$	axiom	
$\vdash \exists x\, \neg\phi(x) \to \neg\phi(c)$	Henkin assumption	
$\vdash \neg\phi(x) \to \neg\phi(c)$	propositional reasoning	
$\vdash \phi(c) \to \phi(x)$	more propositional reasoning	
$\vdash \phi(c) \to \forall x\, \phi(x)$	$\forall$Gen.	QED

7 November 2023

Lemma 1.68. *If T is a consistent L-theory, then there exists a complete consistent L-theory $\tilde{T}$ extending it; i.e., $\tilde{T} \supseteq T$.*

Proof. As in the propositional case, we use Zorn's lemma to show that there exists a maximal consistent L-theory $\tilde{T}$ such that $\tilde{T} \supseteq T$ (recall that the union of a chain of consistent theories is consistent).

To see $\tilde{T}$ is complete, let ϕ be a sentence and suppose $\tilde{T}$ does not prove ϕ. We will show it proves $\neg\phi$.

By maximality of $\tilde{T}$, the theory $\tilde{T} \cup \{\phi\}$ is inconsistent. Then

$$\begin{aligned} \tilde{T}, \phi \vdash \bot &\implies \tilde{T} \vdash \phi \to \bot && \text{deduction theorem} \\ &\implies \tilde{T} \vdash \neg\phi && \text{propositional reasoning.} \quad \text{QED} \end{aligned}$$

Lemma 1.69 (Constants)**.** *Let T be an L-theory, $\phi(x)$ an L-formula, and c a constant symbol such that $c \notin L$. Then*

$$T \vdash \phi(c) \qquad \textit{implies} \qquad T \vdash \phi(x).$$

Proof. Let $\phi_0, \ldots, \phi_n$ be a proof of $\phi(c)$ in T, and y be a variable that does not occur in the proof. Then $\phi_0(y/c), \ldots, \phi_n(y/c)$ is still a valid proof of $(\phi(c/x))(y/c) = \phi(y/x)$ from T. (The meaning of '(y/c)' is that we replace each occurrence of c with y; this is not formally a substitution according to Definition 1.42 as c is not a variable.) Thus, T proves $\phi(y/x)$; we may infer $\forall y\, \phi(y/x)$ using $\forall$Gen, and then $\phi(x)$ using the axiom $\forall y\, \phi(y/x) \to \underbrace{(\phi(y/x))(x/y)}_{\phi(x)}$. QED

Lemma 1.70. *If T is a consistent L-theory, $c \notin L$ a constant symbol, and $\phi(x)$ an L-formula, then the following theory is consistent:*

$$T \cup \{\exists x\, \phi(x) \to \phi(c)\}.$$

Proof. If $T, \exists x\, \phi(x) \to \phi(c) \vdash \bot$, let y be a variable not occurring in $\phi(x)$. Then

$$\begin{aligned} &T \vdash (\exists x\, \phi(x) \to \phi(c)) \to \bot && \text{deduction theorem} \\ &T \vdash (\exists x\, \phi(x) \to \phi(y)) \to \bot && \text{lemma on constants} \\ &T \vdash \exists y\, (\exists x \phi(x) \to \phi(y)) \to \bot && \exists\text{Gen.} \end{aligned}$$

But $\vdash \exists y\, (\exists x\, \phi(x) \to \phi(y))$ (exercise), hence $T \vdash \bot$, which is a contradiction. QED

Lemma 1.71. *Let T be a consistent L-theory; then there exists a language $L_H \supseteq L$ and a consistent Henkin L_H-theory $T_H \supseteq T$.*

Proof. We construct the language L_H and the theory T_H inductively as follows:

$$\begin{aligned} L_0 &= L & L_{n+1} &= L_n \cup \{c_\phi : \phi(x) \text{ is an } L_n\text{-formula}\} \\ T_0 &= T & T_{n+1} &= T_n \cup \{\exists x\, \phi(x) \to \phi(c_\phi) : \phi(x) \text{ is an } L_n\text{-formula}\} \\ L_H &= \bigcup_{n \in \mathbb{N}} L_n & T_H &= \bigcup_{n \in \mathbb{N}} T_n \end{aligned}$$

Note that, despite not writing it out, we introduce (wastefully) a new constant for every L_n-formula for every n, meaning there are formally distinct constants for the same formula considered in the increasingly extended languages.

Clearly, $T_H \supseteq T$. It is, moreover, an L_H-theory. If $\phi(x)$ is an L_H-formula, then $\phi(x)$ is an L_n-formula for some $n \in \mathbb{N}$, hence $T_H \supseteq T_n$ includes the Henkin axiom $\exists x\, \phi(x) \to \phi(c_\phi)$. Thus, T_H is a Henkin theory.

It remains to show that T_H is consistent. It suffices to show that $T_n \nvdash \bot$ for all $n \in \mathbb{N}$. We do this by induction on n. For the base case, $T_0 = T$ is consistent by assumption.

Let us show the induction step for $n+1$. Assume that $T_n \nvdash \bot$, and suppose $T_{n+1} \vdash \bot$ towards a contradiction. Then

$$T_n \cup \{\exists x\, \phi_i(x) \to \phi_i(c_{\phi_i}) \colon i < m\} \vdash \bot$$

for some $m \in \mathbb{N}$ and some L_n-formulas ϕ_i, $i < m$. But this theory is consistent by m applications of Lemma 1.70 (more formally, we should prove this by induction on m). This is a contradiction. QED

Theorem 1.72 (Completeness)**.** *Let Γ be a set of L-formulas and ϕ an L-formula. Then*

$$\Gamma \models \phi \qquad \textit{implies} \qquad \Gamma \vdash \phi.$$

Proof. Assume $\Gamma \nvdash \phi$. Then $\Gamma^\forall \nvdash \phi^\forall$ by Exercise 1.60, thus the theory $T = \Gamma^\forall \cup \{\neg\phi^\forall\}$ is consistent. By Lemma 1.71, T may be extended to a consistent Henkin L_H-theory T_H, which in turn may be extended to a consistent complete L_H-theory $\tilde{T}$ by Lemma 1.68. $\tilde{T}$ remains a Henkin theory.

Since $\tilde{T}$ is a consistent complete Henkin theory, it has a model $\mathcal{A}_H \models \tilde{T}$ by Lemma 1.66. Observe $\mathcal{A}_H$ is, in particular, an L_H-structure.

Let $\mathcal{A}$ be the L-reduct of $\mathcal{A}_H$; i.e., we forget about the interpretations of symbols outside of L.

Then $\mathcal{A} \models T$, whence $\mathcal{A} \models \Gamma^\forall$ and $\mathcal{A} \not\models \phi^\forall$. It follows that $\mathcal{A} \models \Gamma$ and $\mathcal{A} \not\models \phi$, which proves $\Gamma \not\models \phi$. QED

The cardinality of an L-structure is understood to be the cardinality of its underlying set.

Theorem 1.73 (Downward Löwenheim–Skolem theorem)**.** *Let T be an L-theory and $\kappa \geqslant |L|$ an infinite cardinal. If T has a model, then it has a model $\mathcal{A} \models T$ such that $|A| \leqslant \kappa$.*

Proof. Let us estimate the size of the model of T constructed in the proof of Theorem 1.72. Since L-formulas in a language L of cardinality $|L| \leqslant \kappa$ are finite strings made of $\leqslant \kappa$ many possible symbols, there are at most $\kappa^{<\omega} = \kappa$ many

L-formulas. It follows by induction on n that the languages L_n from the proof of Lemma 1.71 satisfy $|L_n| \leqslant \kappa$: for the induction step, we have

$$|L_{n+1}| \leqslant \underbrace{|L_n|}_{\leqslant\kappa} + \underbrace{|\{c_\phi : \phi(x) \text{ is an } L_n\text{-formula}\}|}_{\leqslant\kappa} \leqslant \kappa.$$

This implies $|L_H| \leqslant \kappa$, and in particular, there are $\leqslant \kappa$ closed L_H-terms. Thus, the model of the Henkin completion of T in language L_H defined in the proof of Lemma 1.66 has cardinality at most κ. QED

1.6 Consequences of the completeness theorem

Theorem 1.74 (Compactness)**.** *Let Γ be a set of L-formulas.*

(1) *If $\Gamma \models \phi$, then there is a finite subset $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \models \phi$.*

(2) *Γ has a model iff every finite $\Gamma_0 \subseteq \Gamma$ has a model.*

Proof.

(1) $\Gamma \models \phi$ implies $\Gamma \vdash \phi$ by the Completeness Theorem. By definition, there is a proof $\phi_0, \dots, \phi_n$ of ϕ from Γ. Let $\Gamma_0 = \Gamma \cap \{\phi_0, \dots, \phi_n\}$. Then $\Gamma_0 \subseteq \Gamma$ is finite and $\Gamma_0 \vdash \phi$, thus $\Gamma_0 \models \phi$.

(2) '$\Rightarrow$' is trivial; for '$\Leftarrow$', we apply (1) with $\phi = \bot$. QED

Definition 1.75. If $\mathcal{A}$ is an L-structure, the *(complete) theory of $\mathcal{A}$* is $\mathsf{Th}(\mathcal{A}) = \{\phi : \mathcal{A} \models \phi\}$.

The *standard model of arithmetic* is $\mathbb{N} = \langle \mathbb{N}, 0, 1, +, \cdot, < \rangle$; its theory $\mathsf{Th}(\mathbb{N})$ is called the *true arithmetic*.

Example 1.76. We extend the language of arithmetic L to $L' = L \cup \{c\}$ and put

$$T = \mathsf{Th}(\mathbb{N}) \cup \big\{c > \underbrace{1 + \cdots + 1}_{n} : n \in \mathbb{N}\big\}.$$

Every finite $T_0 \subseteq T$ has a model: if n is the largest such that $c > \overbrace{1 + \cdots + 1}^{n}$ occurs in T_0, then $\langle \mathbb{N}, n+1 \rangle \models T_0$. The compactness theorem then implies T has a model $\mathcal{M}$.

$\mathcal{M}$ is a model of $\mathsf{Th}(\mathbb{N})$, not isomorphic to $\mathbb{N}$. It is clear from the definition that $c^{\mathcal{M}}$ is infinitely large. Note that $\mathbb{N}$ is embedded in $\mathcal{M}$ as an initial segment via the inclusion $n \in \mathbb{N} \mapsto (\underbrace{1 + \cdots + 1}_{n})^{\mathcal{M}}$.

Theorem 1.77 (Löwenheim–Skolem theorem)**.** *Let T be an L-theory and $\kappa \geqslant |L|$ an infinite cardinal. Let us assume that T either has an infinite model, or it has arbitrarily large[2] finite models. Then T has a model of cardinality κ.*

[2] I.e., for every $n \in \mathbb{N}$, T has a model of cardinality at least n.

Proof. The basic idea is to employ κ many constants to ensure that any model has size $\geqslant \kappa$, and apply the downward LS theorem.

Let $L' = L \cup \{c_\alpha : \alpha < \kappa\}$ be new constants and $T' = T \cup \{c_\alpha \neq c_\beta : \alpha < \beta < \kappa\}$. Let us check that every finite $T_0 \subseteq T'$ has a model. T_0 includes axioms $c_\alpha \neq c_\beta$ for $\alpha, \beta \in I$ for some finite I. Let $\mathcal{A} \models T$ be such that $|A| \geqslant |I|$; pick distinct $c_\alpha^{\mathcal{A}} \in A$ for $\alpha \in I$, and pick arbitrary $c_\alpha^{\mathcal{A}} \in A$ for $\alpha \notin I$. Then

$$\left\langle \mathcal{A}, c_\alpha^{\mathcal{A}} : \alpha < \kappa \right\rangle \models T_0.$$

By compactness, T' has a model $\mathcal{A}$. Since $|L'| = \kappa$, we may assume $|A| \leqslant \kappa$ by the downward Löwenheim–Skolem theorem. Because $\mathcal{A} \models T'$, the interpretations $\left\{c_\alpha^{\mathcal{A}} : \alpha < \kappa\right\}$ are pairwise distinct. Thus, $|A| \geqslant \kappa$, whence $|A| = \kappa$. QED

In the previous exercise, we constructed a nonstandard model of arithmetic. The Löwenheim–Skolem theorem tells us that there are such models of arbitrary cardinality.

Definition 1.78. Suppose $\mathcal{A}$ and $\mathcal{B}$ are L-structures for some language L. An *isomorphism* of $\mathcal{A}$ to $\mathcal{B}$ is a bijection $f\colon A \to B$ such that

(1) $R^{\mathcal{A}}(a_0, \ldots, a_{n-1}) \iff R^{\mathcal{B}}(f(a_0), \ldots, f(a_{n-1}))$ for all n-ary relations $R \in L$ and $a_0, \ldots a_{n-1} \in A$,

(2) $F^{\mathcal{B}}(f(a_0), \ldots, f(a_{n-1})) = f(F^{\mathcal{A}}(a_0, \ldots, a_{n-1}))$ for n-ary functions $F \in L$ and $a_0, \ldots, a_{n-1} \in A$.

Definition 1.79. Let T be an L-theory and $\kappa \geqslant \aleph_0$ a cardinal. Then T is *κ-categorical* if all $\mathcal{A} \models T$ of cardinality κ are isomorphic.

Theorem 1.80 (Vaught's test)**.** *If T is a κ-categorical L-theory without finite models, where $\kappa \geqslant |L|$, then T is complete.*

Proof. If T is not complete, there is a sentence ϕ such that $T \cup \{\phi\}$ and $T \cup \{\neg\phi\}$ are consistent. Then there exist models $\mathcal{A} \models T \cup \{\phi\}$, $\mathcal{B} \models T \cup \{\neg\phi\}$, $|A| = |B| = \kappa$ by the Löwenheim–Skolem theorem. We have $\mathcal{A} \models \phi$ and $\mathcal{B} \not\models \phi$, thus $\mathcal{A}$ and $\mathcal{B}$ are not isomorphic. QED

Example 1.81. The theory DLO of dense linear orders without endpoints has language $L = \{<\}$ and the following axioms:

(1) $x \not< x$.

(2) $x < y \wedge y < z \to x < z$.

(3) $x < y \vee x = y \vee y < x$.

(4) $\forall x, y\,(x < y \to \exists z\,(x < z \wedge z < y))$.

(5) $\forall x\,\exists y\, x < y$.

(6) $\forall x\,\exists y\, y < x$.

This theory is $\aleph_0$-categorical, hence complete. This follows by a back-and-forth argument.

Example 1.82. Let $L = \{P(x)\}$. There is no L-sentence ϕ such that for every finite L-structure $\mathcal{M}$,

$$\mathcal{M} \models \phi \iff |P^{\mathcal{M}}| > |\mathcal{M} \smallsetminus P^{\mathcal{M}}|.$$

Assume for contradiction that ϕ is such a sentence, and define

$$T = \left\{ \exists \overline{x} \left(\bigwedge_{i<j<n} x_i \neq x_j \wedge \bigwedge_{i<n} P(x_i) \right), \exists \overline{x} \left(\bigwedge_{i<j<n} x_i \neq x_j \wedge \bigwedge_{i<n} \neg P(x_i) \right) \right\}_{n \in \mathbb{N}}.$$

Every finite $T_0 \subseteq T$ is consistent with ϕ and with $\neg\phi$. By compactness, both $T \cup \{\phi\}$ and $T \cup \{\neg\phi\}$ have models: $\mathcal{A} \models T \cup \{\phi\}$ and $\mathcal{B} \models T \cup \{\neg\phi\}$. We may assume both $\mathcal{A}$ and $\mathcal{B}$ to be countable by the downward Löwenheim–Skolem theorem.

Then writing $\mathcal{A} = \langle A, P^{\mathcal{A}} \rangle$, we have that $A, P^{\mathcal{A}}, A \smallsetminus P^{\mathcal{A}}$ are countably infinite and the same for $\mathcal{B}$. Thus, $\mathcal{A}$ is isomorphic to $\mathcal{B}$, but $\mathcal{A} \models \phi$, $\mathcal{B} \not\models \phi$. This is a contradiction.

Example 1.83. No sentence can define the class of connected graphs.

Theorem 1.84 (De Bruijn–Erdős theorem)**.** *For any $k \in \mathbb{N}$, a graph $G = \langle V, E \rangle$ is k-colourable iff all finite subgraphs G_0 of G are k-colourable.*

Proof. Define

$$\begin{aligned} L_G &= \{E(x,y)\} \cup \{P_i(x) : i < k\} \cup \{\underline{u} : u \in V\}, \\ T_G &= \left\{ \forall x \bigvee_{i<k} P_i(x), \bigwedge_{i<k} \forall x \forall y \, \neg(E(x,y) \wedge P_i(x) \wedge P_i(y)) \right\} \\ &\qquad \cup \{E(\underline{u}, \underline{v}) : \langle u, v \rangle \in E\}. \end{aligned}$$

Then T has a model iff G is k-colourable:

- $\Leftarrow$ Let $c\colon V \to \{0, \ldots, k-1\}$ be a k-colouring of G. Then
$$\mathcal{M} = \langle V, E, P_0^{\mathcal{M}}, \ldots, P_{k-1}^{\mathcal{M}} \rangle_V \models T_G,$$
where $P_i^{\mathcal{M}} = c^{-1}(\{i\})$.
- $\Rightarrow$ Suppose $\mathcal{M} \models T_G$. We define a k-colouring $c\colon V \to \{0, \ldots, k-1\}$ as follows. If $u \in V$, then there is $i < k$ s.t. $\mathcal{M} \models P_i(\underline{u})$; let $c(u)$ be one such i. Then if $\langle u, v \rangle \in E$, then $\mathcal{M} \models E(\underline{u}, \underline{v})$, thus $c(u) \neq c(v)$.

If all finite subgraphs of G are k-colourable, then every finite $T_0 \subseteq T_G$ has a model, as $T_0 \subseteq T_{G_0}$ for some finite $G_0 \subseteq G$. Thus T_G has a model by compactness, whence G is k-colourable. QED

2 Computability

5 December 2023

2.1 Turing machines

We wish to formalize the notion of an effective algorithm. There are several motivations for this.

First, it is intrinsically interesting as effective computability seems to be a fundamental concept for which we would like to have a formal counterpart.

Second, it allows us to mathematically formulate and answer questions about computability of particular problems. If a problem *is* computable, we can show this just by exhibiting an algorithm, for which an intuitive understanding of the concept suffices. However, if we want to prove that some problem is *not* computable, we need a precise definition.

A specific problem suggested by Section 1 is the so-called *Entscheidungsproblem*[3]:

> *Is there an algorithm that would decide whether a given first-order sentence ϕ is logically valid?*

Note that validity of propositional formulas is algorithmically decidable by just trying all assignments (see also Remark 1.24). For first-order sentences, we have a 'one-sided' algorithm: we may systematically enumerate all possible proofs. The completeness theorem ensures that if a sentence *is* valid, we will (in principle) find its proof sooner or later; however, if a sentence is *not* valid, this algorithm will run forever and never halt.

Third, effective algorithms and related concepts are an important tool for investigation of first-order theories of arithmetic, as we will see in Section 3.

Many different formal models for computation have been proposed:

- Turing machines
- General recursive functions
- λ-calculus
- Random-access machines
- ...

However, all of these turned out to be equivalent. This leads to the so-called *Church–Turing thesis*, which posits that a problem is effectively computable in the informal sense iff it is computable by a Turing machine. This is what we will use as our formal model as well.

Intuitively speaking, we mean to formalize the notion of a simple physical device consisting of an internal logic circuitry that can be in finitely many states, with

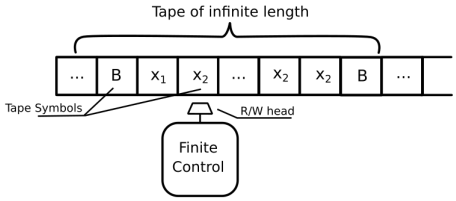
[3] Which literally means 'decision problem' in German, but that term has a much broader meaning in English; see below.

access to a tape divided into discrete cells, each of which can hold one symbol. The tape provides the machine with input, and it is subsequently used as a working memory. The machine can scan the tape using a *reading-writing head* that can move over the tape. We assume the tape has a beginning, but it is infinite in the other direction, and the machine, therefore, may not run out of memory to write into. The formal definition follows:

Definition 2.1. A *Turing machine* is a septuple $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}} \rangle$ comprising:

Set of states Q. This is a finite set containing the set of all 'states' the machine could be in. What the machine does at any given moment is determined by its current 'state' and the input symbol it is reading at that moment (see below for an example).

It also contains three special states denoted $q_0, q_{\text{acc}}, q_{\text{rej}}$; here q_0 denotes the initial state (i.e., the state the machine is in when it is first switched on and a fresh input tape is provided to it—it begins every computation switched into the state q_0), and q_{acc} and q_{rej} stand for the *accepting* and *rejecting* states respectively, which, once the machine switches into either of them, terminate the computation. We assume $q_{\text{acc}} \neq q_{\text{rej}}$.



Input alphabet Σ. This is a finite, nonempty set of symbols the machine recognizes as a valid input. We could consider a finite set of integers, letters, or arbitrary symbols.

Tape alphabet Γ. This is a finite set of symbols that are allowed to be written in cells of the tape. We assume $\Gamma \supseteq \Sigma \cup \{\sqcup\}$, where $\sqcup \notin \Sigma$ is a special symbol called the *blank symbol*. It is the only symbol that may occur infinitely many times on the tape. When the computation begins, all cells save the finitely many containing the actual input string contain the blank symbol.

Transition function δ. The transition function is a function

$$\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}\,.$$

For any given state q and current tape symbol a, it outputs a new (not necessarily distinct) state q' the machine switches itself into, a new (not necessarily distinct) tape symbol a' which the machine writes in the current cell, and one of special commands L or R which tell the machine

whether to move the head one position to the left or to the right (the head is not allowed to stay put in place after the foregoing actions are complete, but this is not an essential restriction).

Definition 2.2. A *configuration* of a Turing machine M is $\langle q, h, u\rangle$ where $q \in Q$ is the *current state*, $h \in \omega$ is the *head position*, and $u \in \Gamma^\omega$ is the *tape content*. We denote the ith cell of u as u_i. A configuration $\langle q, h, u\rangle$ is said to be *accepting* if $q = q_{\text{acc}}$; *rejecting* if $q = q_{\text{rej}}$; and *halting* if it is accepting or rejecting.

It is clear that the transition function uniquely assigns a new configuration to any given non-halting configuration and thus propels the computation. Formally, we say that a configuration $\langle q, h, u\rangle$ such that $q \notin \{q_{\text{acc}}, q_{\text{rej}}\}$ *yields* the configuration $\langle q', h', u'\rangle$ defined as follows. Let $a = u_h$ be the current symbol, and $\delta(q, a) = \langle q', a', t\rangle$. Then $h' = h + 1$ if $t = R$ and $h' = \max(h - 1, 0)$ if $t = L$; $u' \in \Gamma^\omega$ is defined by $u'_h = a'$ and $u'_i = u_i$ for all $i \neq h$.

The *initial configuration* corresponding to input $x \in \Sigma^*$ is $\langle q_0, 0, x{\smile}{\sqcup}^\omega\rangle$. Simply put, the machine is in the initial state, and the head is at the beginning of the tape, whose content is the input string followed by infinitely many blanks.

Definition 2.3 (Run of a Turing machine, acceptance and rejection)**.** A *run* of a Turing machine M on input $x \in \Sigma^*$ is a sequence of configurations $C_0, \ldots, C_t$ where C_0 is the initial configuration on input x, and C_i yields C_{i+1} for each $i < t$.

M is said to *accept*, resp. *reject*, x, if there is a run $C_0, \ldots, C_t$ of M on x where C_t is an accepting, resp. rejecting, configuration.

Remark 2.4. As can be seen from the definitions above, the values of the transition function $\delta(q, a)$ for $q \in \{q_{\text{acc}}, q_{\text{rej}}\}$ are irrelevant, as the machine always halts in such states anyway. Thus, we could have defined δ as only a function $(Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Sigma \to Q \times \Sigma \times \{L, R\}$. We keep the domain to be all of Q for consistency with Sipser's book.

Definition 2.5. A *decision problem* (or *language*; not to be confused with first-order languages) is any subset $L \subseteq \Sigma^*$. That is to say, it is a collection of possible inputs for (some) Turing machine.

Definition 2.6 (Decidability)**.** A Turing machine M is said to *decide*, or *compute*, a decision problem L if for every input $x \in \Sigma^*$:

$$x \in L \implies M \text{ accepts } x, \qquad x \notin L \implies M \text{ rejects } x.$$

A decision problem L is *decidable* (or *computable*, or *recursive*) if there exists a Turing machine M that decides L.

Definition 2.7 (Semidecidability)**.** A Turing machine M is said to *recognize* (or *semidecide*) a decision problem L if for every input $x \in \Sigma^*$ we have

$$x \in L \iff M \text{ accepts } x.$$

Similarly, L is said to be *recognizable* (or *semidecidable*, *computably enumerable*, *recursively enumerable*) if L is recognized by some Turing machine M. The language recognized by M is

$$L(M) = \{x \in \Sigma^* : M \text{ accepts } x\}.$$

Remark 2.8.

- Every Turing machine recognizes exactly one language, viz. $L(M)$. That is, a language L is recognized by a Turing machine M iff $L = L(M)$.
- Perhaps the difference between language recognition and decision ought to be highlighted. A language is recognizable iff there is a Turing machine that will halt and accept only the strings in that language; for strings not in the language, the Turing machine either rejects, or does not halt at all. In contrast, a machine that decides a language must always halt.

  A Turing machine decides a language iff it recognizes it and it halts on every input.

Observation 2.8.1. *Every decidable language is semidecidable.*

Decision problems formalize the notion of computational tasks that admit a YES/NO answer. However, not all problems we might consider computing by an algorithm are of this kind. We will also work with more general problems where the solution can be an arbitrary string; these are called *function problems*[4]:

Definition 2.9 (Partial function)**.** A *partial function* $f\colon X \rightharpoonup Y$ is a function $f\colon \operatorname{dom}(f) \to Y$ such that $\operatorname{dom}(f) \subseteq X$; i.e., f is possibly defined only on a portion of X and not necessarily everywhere. In this context, a function is said to be *total* if $\operatorname{dom}(f) = X$; i.e., the usual notion of *a* function.

Definition 2.10 (Function problem)**.** A *function problem* is partial function $f\colon \Sigma^* \rightharpoonup \Sigma^*$.

Definition 2.11 (Function-problem computation)**.** A Turing machine M is said to *output* $y \in \Sigma^*$ on input $x \in \Sigma^*$ if there is an accepting run $C_0, \ldots, C_t$ of M on input x such that $C_t = \langle q_{\text{acc}}, h, y \underset{\smile}{\sqcup}^{\omega} \rangle$.

M is then said to *compute* a function problem $f\colon \Sigma^* \rightharpoonup \Sigma^*$ if $L(M)$ is the domain of f and for each $x \in \operatorname{dom}(f)$, M outputs $f(x)$ on input x.

$f\colon \Sigma^* \rightharpoonup \Sigma^*$ is a *partial computable function* (or *partial recursive function*) if there is a Turing machine M that computes it. A partial computable function f that is defined everywhere on Σ^* (i.e., it is total) is said to be simply *computable* (or *recursive*).

[4] Even more generally, we could consider problems that admit more than one valid solution; these are called *search problems*, and are important in computational complexity, but we will not see them in this course.

Remark 2.12. We note there are many variants on the definitions of a Turing machine in the literature. Some of the common modifications include:

- A two-sided infinite tape.
- Considering partial transition functions with no rejecting state.
- Multi-tape Turing machines (so-called k-tape machines): the machine has k tapes (where k is a fixed number that's part of the specification of the machine) including an *input tape* (usually read-only), several *work tapes*, and if we are interested in function problems, an *output tape* (usually write-only); each tape has its own head that can move (or stay put) independently.

Exercise 2.13. A k-tape Turing machine is equivalent to a single-tape Turing machine.

Hint: The idea is to represent the content of all k tapes on one tape using a new tape alphabet

$$(\Gamma \times \{0,1\})^k \cup \{\sqcup\}\,.$$

Remark 2.14. Another variation of the definition of Turing machines is to require that $\Gamma = \Sigma \cup \{\sqcup\}$. If $|\Sigma| \geqslant 2$, this can be shown equivalent to the original definition as follows. Given a machine

$$M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{\text{acc}}, q_{\text{rej}} \rangle\,,$$

we fix k such that $|\Gamma| \leqslant (|\Sigma|+1)^k$, and an injective encoding $e\colon \Gamma \to (\Sigma \cup \{\sqcup\})^k$. We may assume $e(\sqcup) = \sqcup^k$. We simulate M using k-tuples of symbols from $\Sigma \cup \{\sqcup\}$ to represent each symbol on the tape using e.

We need to expand the input to the tape encoding before the simulation. This may be done symbol-wise: take a symbol a that had not been encoded yet, shift the content of the tape to the right of the symbol by $k-1$ positions to make room, go back to write the encoding $e(a)$, and repeat until we get to the end of the original input.

Example 2.15. We consider the language

$$\textsc{Palindromes} = \left\{w \in \{a,b\}^* : w = w^R\right\},$$

where the string reversal operator R is defined by $(w_0 \dots w_{n-1})^R = w_{n-1} \dots w_0$. In other words, the language consists of words over a 2-letter alphabet $\{a,b\}$ that are written the same way forwards and backwards; e.g., *bab* or *abba*. We will now design a Turing machine that decides whether a given word is a palindrome.

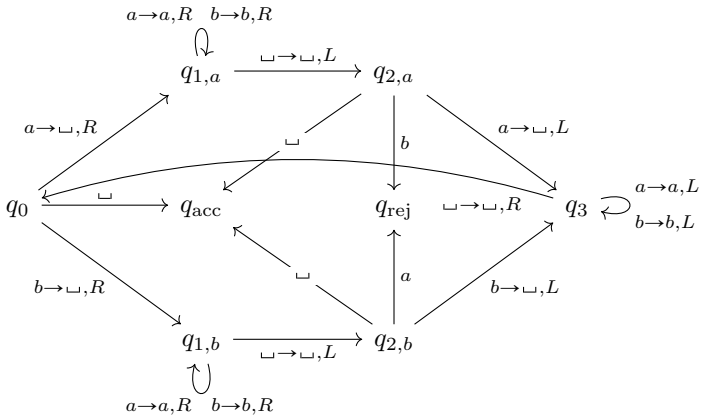
A simple algorithm is to repeatedly check that both symbols at the ends of the string are the same and cross them out, until we either detect an inconsistency or end up with a string of length $\leqslant 1$. A Turing machine cannot operate at both ends simultaneously, but we can achieve something similar by moving the head back and forth: the machine removes the left-most symbol and 'remembers' it

in its internal state, moves to the right end, checks that the last symbol agrees with the remembered one and removes it, and rewinds back to the left.

Formally, we define the machine as

$$M = \langle Q, \{a, b\}, \{a, b, \sqcup\}, \delta, q_0, q_{\text{acc}}, q_{\text{rej}} \rangle,$$
$$Q = \langle q_0, q_{\text{acc}}, q_{\text{rej}}, q_{1,a}, q_{1,b}, q_{2,a}, q_{2,b}, q_3 \rangle,$$

where the transition function δ is given by the following diagram:



Palindromes can be recognized more easily on a two-tape Turing machine: we can copy the string to a work tape and then traverse the two copies in opposite directions.

Remark 2.16. Programming Turing machines down to an explicit listing of the transition function table can be a tedious endeavour that requires a lot of determination and patience, while the result is not very illuminating and obscures the ideas behind the algorithm. The purpose of Example 2.15 is to present during the lecture at least once a complete Turing machine with all the bells and whistles that computes something sensible to show that it can be done indeed, but from now on we will rather describe Turing machines using an informal pseudo-code, assuming that the reader can imagine how to translate it to a formal presentation if required.

If desired—to get a better feeling for what can be implemented on Turing machines and how, or just for fun—there are a number of online Turing machines simulators one can play with, e.g., `https://turingmachinesimulator.com`. The palindrome machine from Example 2.15 can be found at `https://turingmachinesimulator.com/shared/slylqbjruc`.

12 December 2023

Lemma 2.17. *A language $L \subseteq \Sigma^*$ is decidable iff L and $\Sigma^* \setminus L$ are semidecidable.*

Proof.

⇒ Suppose L is decidable; then so is $\Sigma^* \setminus L$ (we may take a Turing machine deciding L and swap the accepting and rejecting states). It follows $\Sigma^* \setminus L$ is semidecidable, while L itself is semidecidable trivially.

⇐ Let M_0 recognize (semidecide) L, and M_1 recognize $\Sigma^* \setminus L$. A 2-tape Turing machine M described by the following pseudo-code decides L:

```
(1) Copy input onto the second tape.
(2) Run M_0 and M_1 in parallel on the two tapes.
(3) If M_0 accepts, ACCEPT.
(4) If M_1 accepts, REJECT.
```

Observe that M has to eventually halt by definition of language recognition: clearly, the input x belongs to either Σ^* or its complement $\Sigma^* \setminus L$. In case of the former, M_0 accepts x by definition; and in case of the latter, M_1 does. Hence x is accepted by precisely one machine and the algorithm halts in finite time. QED

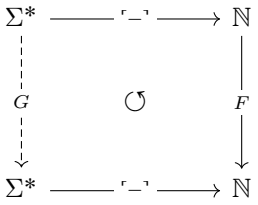
Definition 2.18 (Numbering of strings)**.** Let $k \geqslant 2$, and assume Σ is the finite alphabet $\{1, 2, 3, \ldots, k\}$. Then we may define the following encoding (so-called *bijective base-k numeration*):

$$\ulcorner - \urcorner \colon \Sigma^* \to \mathbb{N}, \qquad \ulcorner a_0 \ldots a_{n-1} \urcorner = \sum_{i<n} k^i a_i.$$

Observe $\ulcorner - \urcorner$ is a bijection.

Definition 2.19. We say that $L \subseteq \mathbb{N}$ is *(semi)decidable* if its bijective base-2 encoding $\left\{w \in \{1,2\}^* : \ulcorner w \urcorner \in L\right\}$ is (semi)decidable.

We say $F \colon \mathbb{N} \rightharpoonup \mathbb{N}$ is *computable* if $G \colon \{1,2\}^* \rightharpoonup \{1,2\}^*$ is computable, where G is uniquely determined by $F(\ulcorner w \urcorner) = \ulcorner G(w) \urcorner$. In other words, G is given by the following commutative diagram:



Definition 2.20. We say $R \subseteq (\Sigma^*)^k$ is *(semi)decidable* if

$$\{w_0 \# w_1 \# \cdots \# w_{k-1} : \langle w_0, \ldots, w_{k-1} \rangle \in R\} \subseteq (\Sigma \cup \{\#\})^*$$

is semidecidable, where $\# \notin \Sigma$ is a new separator symbol.

Similarly, $F \colon (\Sigma^*)^k \rightharpoonup \Sigma^*$ is *computable* if the function $G \colon (\Sigma \cup \{\#\})^* \rightharpoonup \Sigma^*$, $G(w_0 \# w_1 \# \cdots \# w_{k-1}) = F(w_0, \ldots, w_{k-1})$, is computable.

2.2 Universal Turing machines and the halting problem

Warning. Angle brackets $\langle - \rangle$ are used for multiple purposes below, including to denote ordered tuples.

Theorem 2.21. *Let Σ be an alphabet with at least two symbols. Then there exists a* universal Turing machine U_Σ *with the following property:*

For every Turing machine M on the same alphabet Σ, there is a code $\langle M \rangle \in \Sigma^*$ such that

$$U_\Sigma(\langle M \rangle x) \simeq M(x) \qquad \text{for each } x \in \Sigma^*.$$

Here, '$\simeq$' means that U_Σ accepts $\langle M \rangle x$ iff M accepts x, and it rejects $\langle M \rangle x$ iff M rejects x. Moreover, U_Σ outputs $y \in \Sigma^*$ on input $\langle M \rangle x$ iff M outputs y on input x.

Proof. Let $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}} \rangle$ be a Turing machine on the alphabet Σ, where we assume $\Gamma = \Sigma \cup \{\sqcup\}$ (see Remark 2.14). We fix an enumeration $Q = \{q_i : i < s\}$, where q_0 the initial state as indicated above, $q_1 = q_{\text{acc}}$, and $q_2 = q_{\text{rej}}$. We also fix an enumeration $\Gamma = \{a_j : j < k\}$.

The code $\langle M \rangle$ of M will describe the transition function $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$. For convenience, we will use some auxiliary extra symbols ($\#$, $\underline{\#}$, 0, 1, L, R) to define $\langle M \rangle$ and the operation of U_Σ (also, $\langle M \rangle$ may include blanks). Officially, U_Σ is required to have input alphabet Σ, and in particular, we should make $\langle M \rangle \in \Sigma^*$; we achieve this by encoding the expanded alphabet $\Sigma' = \Sigma \cup \{\sqcup, \#, \underline{\#}, 0, 1, L, R\}$ by c-tuples of symbols from Σ for a suitable c, similarly to Remark 2.14. (This is where we use the assumption $|\Sigma| \geqslant 2$.)

We define

$$\begin{aligned}\langle M \rangle = \#\#\langle \delta(q_0, a_0)\rangle \# \langle \delta(q_0, a_1)\rangle \# \cdots \# \langle \delta(q_0, a_{k-1})\rangle \\ \# \langle \delta(q_1, a_0)\rangle \# \langle \delta(q_1, a_1)\rangle \# \cdots \# \langle \delta(q_1, a_{k-1})\rangle \\ \vdots \\ \# \langle \delta(q_{s-1}, a_0)\rangle \# \cdots \# \langle \delta(q_{s-1}, a_{k-1})\rangle \#\#\end{aligned}$$

where if $\delta(q_i, a_j) = \langle q_{i'}, a_{j'}, t \rangle \in Q \times \Sigma \times \{L, R\}$, we define the encoding of $\delta(q_i, a_j)$ as

$$\langle \delta(q_i, a_j) \rangle = a_{j'} t 0^{i'}.$$

Here, $0^{i'}$ denotes a string of zeroes of length i'.

We shall now describe the operation of the universal Turing machine U_Σ. It maintains on its tape a representation of the current configuration of M; it works in an endless loop where on each iteration, it simulates the effects of one step of M. It will be convenient for this purpose to represent a configuration of M as

$$u_0 \dots u_{h-1} \langle M \rangle u_h u_{h+1} \dots$$

where $u_0 \dots u_{h-1}$ is the content of the simulated tape to the left of the head position h, and $u_h u_{h+1} \dots$ the rest of the tape from the head position onward. During the simulation, some parts of $\langle M \rangle$ will be modified a little; we will still refer to it as $\langle M \rangle$. In particular, we need to indicate the current state q_i of M: we do this by replacing the # in front of the entry $\langle \delta(q_i, a_0) \rangle$ of the encoded transition function table (i.e., the beginning of the row of the table corresponding to q_i) with the symbol $\underline{\#}$.

Note that the encoding of Σ' by Σ^c is applied only to the $\langle M \rangle$ part of the configurations; the symbols u_i of the simulated tape will be written literally.

During the simulation, the head will be kept inside $\langle M \rangle$, except possibly venturing one step outside to read/write the current symbol of the simulated machine, or to move the simulated head. We can rewind the tape to the left or right end of $\langle M \rangle$ at any time, because these can be recognized by the substring ## (or $\#\underline{\#}$); this works even after the encoding of Σ' by Σ^c because we never go far away from $\langle M \rangle$, and therefore we cannot lose track of whether we are currently seeing an encoded symbol of Σ' or an unencoded symbol of Σ.

In the beginning of the simulation, U_Σ starts with $\langle M \rangle x$ on the input tape, which is almost a valid representation of the initial configuration of M on input x—we only need to mark the row corresponding to the initial state q_0:

(00) Move right and replace the second # with $\underline{\#}$.

Next comes the main loop of U_Σ, simulating one step of the computation of M. Suppose that the tape contains a representation of a configuration as above. We first have to locate the entry of the transition table corresponding to the current state q_i and symbol under the head a_j of the simulated machine:

(01) Move past the right end of $\langle M \rangle$.
(02) Read and remember $u_h = a_j$.
(03) Locate the $\underline{\#}$ symbol, and replace it with #.
(04) Repeat j times: move right towards the next # symbol.

The head is now at the # symbol in front of the string $\langle \delta(q_i, a_j) \rangle = a_{j'} t 0^{i'}$.

(05) Read and remember $a_{j'}$ and t.

We now have to mark the table row corresponding to the new state $q_{i'}$ with $\underline{\#}$. Unlike $a_{j'}$ or t, we cannot just read it and remember in the state of U_Σ, because the number of states of M may be arbitrarily large (it is not bounded by a constant). We proceed as follows: we convert the $0^{i'}$ in $\langle \delta(q_i, a_j) \rangle$ to $1^{i'}$, mark the row corresponding to q_0, and then use a loop that converts the 1's back to 0's one by one, each time moving the marker to the next row.

However, we must take care to abort the simulation if the new state is halting. Note that we *can* count up to 2 in the state of U_Σ, thus we can check if $i' = 1$ or $i' = 2$, even if we cannot remember an arbitrarily large i'. (This is the reason we fixed the accepting and rejecting states to be q_1 and q_2, resp.)

(06) Move right to the next #, replacing 0 with 1 as we go.
(07) If the total number of 0s was 1, then ACCEPT.
(08) If the total number of 0s was 2, then REJECT.
(09) Locate the left end of $\langle M \rangle$.
(10) Change the second # to $\underline{\#}$.

Now comes the loop for moving the marker. Note that each row of the table has $k = |\Gamma| = |\Sigma| + 1$ entries, which is a constant that we can count up to in the state of U_Σ.

(11) Move right to locate the first 1.
(12) If none is found before the end of $\langle M \rangle$, go to (18).
(13) Change the 1 to 0.
(14) Locate the $\underline{\#}$ symbol and change it to #.
(15) Move to the kth # to the right.
(16) Change it to $\underline{\#}$.
(17) Move to the beginning of $\langle M \rangle$, and go to (11).

We have now placed $\underline{\#}$ correctly to mark the new state. We have yet to update the symbol under the simulated head, and move the head. Note that upon exiting the loop above, the head of U_Σ is past the right end of $\langle M \rangle$, i.e., at the position of u_h.

(18) If $t = R$:
(19) Shift $\langle M \rangle$ to the right (overwriting u_h).
(20) Write $a_{j'}$ in the free space to the left of it.
(21) If $t = L$:
(22) Write $a_{j'}$.
(23) Locate the left end of $\langle M \rangle$.
(24) If it is at the beginning of the tape, go to (01).
(25) Remember the symbol $u_{h-1} = a_{j''}$ to the left of it.
(26) Shift $\langle M \rangle$ to the left (overwriting u_{h-1}).
(27) Write $a_{j''}$ in the free space to the right.
(28) Go to (01).

This finishes the simulation for decision problems. If we care about computation of functions, U_Σ cannot literally halt in step (07): it must first clean up the tape (i.e., remove $\langle M \rangle$ and shift $u_h u_{h+1} \dots$ accordingly) so that its output is the same as the output of M. QED

Definition 2.22. The *halting problem* for a given alphabet Σ is the language

$$A_\Sigma = \{\langle M \rangle x : M \text{ accepts } x\} \subseteq \Sigma^*.$$

Theorem 2.23. *The halting problem A_Σ is semidecidable, but not decidable.*

Proof. A_Σ is recognized by the universal Turing machine U_Σ. To see it is not decidable, we assume towards a contradiction that A_Σ *is* decided by some Turing

machine H. This means that H accepts the pair $\langle M\rangle x$ if M accepts x, and rejects it otherwise.

We define a new Turing machine D on the input alphabet Σ that works as follows:

(1) Duplicate the input x to xx.
(2) Run H.
(3) If H accepts, reject; if H rejects, accept.

We run D on input $\langle D\rangle$, and obtain a contradiction:

$$\begin{aligned} D \text{ accepts } \langle D\rangle &\iff H \text{ rejects } \langle D\rangle\langle D\rangle && \text{definition of } D \\ &\iff \langle D\rangle\langle D\rangle \notin A_\Sigma && H \text{ decides } A_\Sigma \\ &\iff D \text{ does not accept } \langle D\rangle && \text{definition of } A_\Sigma. \end{aligned}$$

QED

Remark 2.24. D stands for 'diagonal machine', as the proof of Theorem 2.23 is a variant of Cantor's diagonal argument. We imagine the infinite matrix indexed by strings where rows enumerate codes $\langle M\rangle$ of Turing machines, columns enumerate inputs x, and a 0/1 entry in the matrix indicates whether M accepts x or not. The machine D computes the diagonal of the matrix with 0/1 flipped, but this clearly cannot agree with any row of the matrix.

Once we identified one undecidable problem, we can prove the undecidability of other problems by means of reductions:

Definition 2.25. Let $A, B \subseteq \Sigma^*$. We say A is *many-one reducible* (or *mapping reducible*) to B, written $A \leqslant_m B$, if there is a computable $f\colon \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$ we have

$$x \in A \qquad \iff \qquad f(x) \in B.$$

We say A and B are *many-one equivalent*, written $A \equiv_m B$, if $A \leqslant_m B$ and $B \leqslant_m A$.

Lemma 2.26. *If $A \leqslant_m B$ and B is (semi)decidable, then A is (semi)decidable.*

Proof. If M (semi)decides B, and M' computes f, then A is (semi)decided by the Turing machine that simulates M' to compute $f(x)$ and then simulates M. QED

Exercise 2.27. We defined the 'halting problem' to be A_Σ in accordance with Sipser's book, but it would be more logical to call A_Σ the 'acceptance problem', and reserve the name 'halting problem' for $H_\Sigma = \{\langle M\rangle x : M \text{ halts on input } x\}$ (which is the traditional definition). Fortunately, this does not make a significant difference: prove that $H_\Sigma \equiv_m A_\Sigma$.

We will see some more serious applications of reductions to proving undecidability in Section 3.

3 Arithmetic

19 December 2023

3.1 Robinson and Peano arithmetic

We wish to axiomatize the structure $\mathbb{N} = \langle \mathbb{N}, 0, S, +, \cdot, \leqslant \rangle$. One way of defining $\mathbb{N}$ up to isomorphism are the *Dedekind–Peano axioms*. They postulate:

(1) There is a natural number 0.

(2) There is a successor function S on $\mathbb{N}$. The successor of any natural number is a natural number.

(3) No two distinct natural numbers have a common successor.

(4) The number 0 is the successor of no natural number.

(5) A set of natural numbers containing 0 which is closed under S is the set of all natural numbers.

Observe that these axioms do not form a first-order theory since the last axiom quantifies over subsets. One might think we could amend this by considering a structure with domain $\mathcal{P}(\mathbb{N})$, but this does not really work either because for any theory in such a language, there is no way of enforcing that its model includes *all* subsets of $\mathbb{N}$ using first-order axioms.

After all, we know from Section 1 that any first-order theory of arithmetic will have nonstandard models of arbitrarily large cardinality, hence there is no way it could define $\mathbb{N}$ up to isomorphism; but we may still hope to capture all first-order sentences valid in $\mathbb{N}$ by a nice explicit set of natural axioms. (This is possible for some related structures, such as $\langle \mathbb{N}, 0, 1, +, \leqslant \rangle$ and $\langle \mathbb{R}, 0, 1, +, \cdot, \leqslant \rangle$.) A natural attempt at such an axiomatization is *Peano arithmetic*.

Definition 3.1 (Robinson and Peano arithmetic)**.** The *language of arithmetic* is $L_{\mathsf{PA}} = \{0, S, +, \cdot, \leqslant\}$. *Robinson's arithmetic* Q is the L_{PA}-theory with axioms

Q1 $S(x) = S(y) \to x = y$,

Q2 $S(x) \neq 0$,

Q3 $x \neq 0 \to \exists y\, S(y) = x$,

Q4 $x + 0 = x$,

Q5 $x + S(y) = S(x + y)$,

Q6 $x \cdot 0 = 0$,

Q7 $x \cdot S(y) = x \cdot y + x$,

Q8 $x \leqslant y \leftrightarrow \exists z\, z + x = y$.

Peano arithmetic PA is Q extended with the schema of induction

$$\phi(0) \wedge \forall x \left(\phi(x) \to \phi(S(x))\right) \to \forall x\, \phi(x) \qquad \text{for all formulas } \phi.$$

PA is a first-order version of the Dedekind–Peano axioms, and may look as a plausible candidate for a complete axiomatization of true arithmetic $\mathsf{Th}(\mathbb{N})$. However, we will prove that it is in fact incomplete, and it cannot be made complete by adding any semidecidable set of axioms; furthermore, this holds already for extensions of the rudimentary theory Q. This is the content of Gödel's first incompleteness theorem.

We will derive the *incompleteness* of extensions of Robinson's arithmetic from their *undecidability*. Towards that goal, we will show that we can 'represent' semidecidable sets $X \subseteq \mathbb{N}$ in the theory. We do this in two steps:

- Semidecidable sets are definable by so-called Σ_1-formulas in $\mathbb{N}$.
  - The main technical ingredient here is that L_{PA} is expressible enough to define encoding of finite sequences.
- Σ_1-sentences true in $\mathbb{N}$ are provable in Q.

We start with the second bullet point.

3.2 Σ_1-completeness of Q

Definition 3.2. By *bounded quantifiers* we understand the abbreviations

$$\exists x \leqslant t\, \phi \equiv \exists x\, (x \leqslant t \wedge \phi), \qquad \forall x \leqslant t\, \phi \equiv \forall x\, (x \leqslant t \to \phi),$$

where t is a term not containing x.

A formula is said to be *bounded*, or Δ_0, if all its quantifiers are bounded.

A formula $\phi(\overline{x})$ is Σ_1 if it has the form $\exists \overline{y}\, \theta(\overline{x}, \overline{y})$ where θ is bounded.

Definition 3.3. A Δ_0-*function* is a partial function $f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ satisfying the following two conditions:

(1) There is a term t (i.e., a polynomial with coefficients from $\mathbb{N}$) such that

$$\forall \overline{x} \in \mathbb{N}^k\, f(\overline{x}) \leqslant t(\overline{x}).$$

(2) There is a Δ_0-formula $\theta(\overline{x}, y)$ such that

$$f(\overline{x}) = y \iff \mathbb{N} \models \theta(\overline{x}, y).$$

Definition 3.4. The *numeral* representing $n \in \mathbb{N}$ is the closed term

$$\overline{n} = \underbrace{S(S(\cdots(S}_{n \text{ times}}(0))\cdots)).$$

Formally, we define $\overline{n}$ by induction (in the meta-theory) as $\overline{0} = 0$ and $\overline{n+1} = S(\overline{n})$. It follows from the definition that

$$\overline{n}^{\mathbb{N}} = n.$$

Lemma 3.5. *Let $n, m \in \mathbb{N}$. Then*

(1) $\mathsf{Q} \vdash \overline{n} + \overline{m} = \overline{n+m}$.

(2) $\mathsf{Q} \vdash \overline{n} \cdot \overline{m} = \overline{nm}$.

(3) $\mathsf{Q} \vdash \overline{n} \neq \overline{m}$ *if* $n \neq m$.

(4) $\mathsf{Q} \vdash \forall x\,(x \leqslant \overline{n} \leftrightarrow x = \overline{0} \vee x = \overline{1} \vee \cdots \vee x = \overline{n})$.

Proof.

(1) By induction (in the meta-theory) on m. The base case $m = 0$ is clear:

$$\mathsf{Q} \vdash \overline{n} + 0 \overset{\text{Q4}}{=} \overline{n} = \overline{n+0}.$$

For the induction step $m \mapsto m+1$:

$$\mathsf{Q} \vdash \overline{n} + \overline{m+1} \overset{\text{def}}{=} \overline{n} + S(\overline{m}) \overset{\text{Q5}}{=} S(\overline{n} + \overline{m}) \overset{\text{i.h.}}{=} S(\overline{n+m}) \overset{\text{def}}{=} \overline{n+m+1}.$$

(2) Again, the proof is by meta-induction on m. The base case $m = 0$ is Q6. For the induction step, Q proves

$$\overline{n} \cdot \overline{m+1} \overset{\text{def}}{=} \overline{n} \cdot S(\overline{m}) \overset{\text{Q7}}{=} \overline{n} \cdot \overline{m} + \overline{n} \overset{\text{i.h.}}{=} \overline{nm} + \overline{n} \overset{(1)}{=} \underbrace{\overline{nm+n}}_{n(m+1)}.$$

(3) By meta-induction on $\min\{n, m\}$. First suppose $m = 0 < n$. Then

$$\mathsf{Q} \vdash \overline{n} \overset{\text{def}}{=} S(\overline{n-1}) \overset{\text{Q2}}{\neq} 0.$$

Similarly if $m > 0 = n$. Finally, if $n, m > 0$, we have

$$\begin{aligned} &\mathsf{Q} \vdash \overline{n} = \overline{m} \to \overline{n-1} = \overline{m-1} && \text{Q1,} \\ &\mathsf{Q} \vdash \overline{n-1} \neq \overline{m-1} && \text{induction hypothesis.} \end{aligned}$$

(4) ($\leftarrow$) If $m \leqslant n$, then

$$\begin{aligned} &\mathsf{Q} \vdash \overline{n} = \overline{n-m} + \overline{m} && \text{by (1),} \\ &\quad \vdash \overline{m} \leqslant \overline{n} && \text{by Q8.} \end{aligned}$$

($\rightarrow$) By meta-induction on n:

Base case $n = 0$. Let us reason in Q. If $x \leqslant 0$, then $z + x = 0$ for some z by Q8. By Q3, either $x = 0$ and we are done, or $x = S(y)$ for some y. But then $0 = z + S(y) = S(z + y)$ by Q5, contradicting Q2.

Induction step $n \mapsto n+1$**.** Reason in Q, and assume $x \leqslant \overline{n+1}$. Again, we have $\overline{n+1} = z + x$ for some z by Q8, and either $x = 0$ (in which case we are done) or $x = S(y)$ for some y.

In the latter case, $S(z+y) = \overline{n+1} = S(\overline{n})$ by Q5, thus Q1 implies $z + y = \overline{n}$, i.e., $y \leqslant \overline{n}$ by Q8.

Then y is 0 or $\overline{1}$ or ... or $\overline{n}$ by the induction hypothesis, thus x is $\overline{1}$ or $\overline{2}$ or ... or $\overline{n+1}$. QED

Lemma 3.6. *If t is a closed L_{PA}-term, and $t^{\mathbb{N}} = n$, then $\mathsf{Q} \vdash t = \overline{n}$.*

Proof. By induction on the complexity of t using (1) and (2) of Lemma 3.5. QED

Lemma 3.7. *Let θ be a Δ_0 sentence. Then*

$$\mathbb{N} \models \theta \implies \mathsf{Q} \vdash \theta,$$
$$\mathbb{N} \not\models \theta \implies \mathsf{Q} \vdash \neg\theta.$$

Proof. By induction on the complexity of θ:

Atomic formulas. Suppose θ is $t = s$ or $t \leqslant s$ for some closed terms t and s (they have to be closed as θ is a sentence). Let $n = t^{\mathbb{N}}$ and $m = s^{\mathbb{N}}$. By the previous lemma, $\mathsf{Q} \vdash t = \overline{n}$ and $\mathsf{Q} \vdash s = \overline{m}$. Moreover, using Lemma 3.5,

$$\begin{aligned}
&\mathbb{N} \models t = s \implies n = m \implies \mathsf{Q} \vdash \overline{n} = \overline{m}, && \\
&\mathbb{N} \not\models t = s \implies n \neq m \implies \mathsf{Q} \vdash \overline{n} \neq \overline{m} && \text{by (3)}, \\
&\mathbb{N} \models t \leqslant s \implies n \leqslant m \implies \mathsf{Q} \vdash \overline{n} \leqslant \overline{m} && \text{by (4)}, \\
&\mathbb{N} \models t \not\leqslant s \implies n \not\leqslant m \implies \mathsf{Q} \vdash \overline{n} \not\leqslant \overline{m} && \text{by (4) and (3)}.
\end{aligned}$$

Conjunction, Disjunction, Negation. This case is left as an exercise.

Universal quantification. Suppose $\theta = \forall x \leqslant t\, \theta_0(x)$ for some closed term t. As before, we have $\mathsf{Q} \vdash t = \overline{n}$, where $n = t^{\mathbb{N}}$.

- Suppose $\mathbb{N} \models \theta$. It follows that for each $m \leqslant n$, we have $\mathbb{N} \models \theta_0(\overline{m})$, thus $\mathsf{Q} \vdash \theta_0(\overline{m})$ by the induction hypothesis. Moreover,

$$\mathsf{Q} \vdash x \leqslant t \to x = \overline{0} \vee x = \overline{1} \vee \cdots \vee x = \overline{n}$$

by (4), thus

$$\mathsf{Q} \vdash x \leqslant t \to \theta_0(x),$$

whence $\mathsf{Q} \vdash \forall x \leqslant t\, \theta_0(x)$ by Remark 1.57.

- Suppose $\mathbb{N} \not\models \theta$. Then there is $m \leqslant n$ such that $N \not\models \theta_0(\overline{m})$; whence by the induction hypothesis, $\mathsf{Q} \vdash \neg\theta_0(\overline{m})$. Moreover, $\mathsf{Q} \vdash \overline{m} \leqslant \overline{n} = t$ by (4), hence $\mathsf{Q} \vdash \neg\forall x \leqslant t\, \theta_0(x)$.

Existential quantification. This case is left as an exercise. QED

Theorem 3.8 (Σ_1-completeness)**.** *Every true*[5] *Σ_1-sentence ϕ is provable in Robinson's arithmetic* Q*.*

Proof. Let $\phi = \exists x_0, \dots, x_{k-1}\, \theta(x_0, \dots)$ be a Σ_1-sentence, where $\theta \in \Delta_0$. Then $\mathbb{N} \models \phi$ implies there are some $n_0, \dots, n_{k-1} \in \mathbb{N}$ such that $\mathbb{N} \models \theta(n_0, \dots)$, which implies $\mathsf{Q} \vdash \theta(\overline{n_0}, \dots)$ by the previous lemma, whence $\mathsf{Q} \vdash \exists x_0, \dots\, \theta(x_0, \dots)$. QED

3.3 Sequence encoding and definability of computation

Definition 3.9. In the context of arithmetic, we will use the pairing function $\langle x, y\rangle = (x + y)^2 + x$.

The next lemma states that it is indeed a pairing function:

Lemma 3.10. $\mathbb{N} \models \forall x, y, u, v\, (\langle x, y\rangle = \langle u, v\rangle \to x = y \wedge u = v)$.

Proof. We have $(x + y)^2 \leqslant \langle x, y\rangle < (x + y + 1)^2$. Thus, if $\langle x, y\rangle = \langle u, v\rangle$, then $(x + y)^2 = (u + v)^2$, which implies $x + y = u + v$, which implies $x = u$, whence $y = v$. QED

Theorem 3.11 (Sequence encoding)**.** *There exists a Δ_0-function $\beta(x, m, i)$ such that for any $k \in \mathbb{N}$ and $x_0, \dots, x_{k-1} \in \mathbb{N}$, there are $x, m \in \mathbb{N}$ that encode this sequence via β:*

$$\beta(x, m, i) = x_i \qquad \text{for all } i < k.$$

Definition 3.12. Given $x, y \in \mathbb{N}$ and $y > 0$, $\operatorname{rem}(x, y)$ is the unique r such that $0 \leqslant r < y$ and $x \equiv r \pmod{y}$. That is, $\operatorname{rem}(x, y) = x - y\lfloor x/y\rfloor$.

We define $\beta(x, m, i) = \operatorname{rem}(x, 1 + (i + 1)m)$. This is called *Gödel's β-function.* 9 January 2023

Lemma 3.13. *If $1, \dots, k \mid m$, then $\{1 + im : i \leqslant k\}$ are pairwise coprime.*

Proof. If $i < j \leqslant k$ and $p \mid 1 + im, 1 + jm$ is prime, then $p \mid (i - j)m$ implies $p \mid m^2$, whence $p \mid m$ and thus $p \mid 1$. A contradiction. QED

Lemma 3.14 (Chinese remainder theorem)**.** *If $m_0, m_1, \dots, m_{k-1}$ are pairwise coprime natural numbers, then for all $x_0, x_1, \dots, x_{k-1} \in \mathbb{N}$, there exists $x \in \mathbb{N}$ such that*

$$x \equiv x_i \pmod{m_i} \qquad \text{for all } i < k.$$

[5] I.e., true in the standard model $\mathbb{N}$.

Proof of Theorem 3.11. We have $\beta(x, m, i) \leqslant x$, and the graph of β is definable by the Δ_0-formula

$$\beta(x, m, i) = y \iff \exists q \leqslant x\, x = y + q \cdot (1 + (1 + i) \cdot m).$$

Thus, β is a Δ_0-function.

Let k and $x_0, \dots, x_{k-1}$ be given. Fix some m such that $1, \dots, k \mid m$ and $m \geqslant x_i$ for all $i < k$. Since $1 + m, \dots, 1 + km$ are coprime by Lemma 3.13, the Chinese remainder theorem tells us there is some x such that

$$x \equiv x_i \pmod{1 + (1 + i)m} \qquad \text{for all } i < k.$$

Also $x_i < 1 + (1 + i)m$, thus $x_i = \operatorname{rem}(x, 1 + (1 + i)m) = \beta(x, m, i)$. QED

As defined, Gödel's β-function needs a pair of numbers x, m to encode a sequence $x_0, \dots, x_{k-1}$, and even so it does not determine the length of the sequence (k). Thus, we introduce a variant of the function that is more convenient to use:

Definition 3.15.

$$\begin{aligned} \operatorname{seq}(w, i) = y &\iff \exists x, m, k \leqslant w\, (w = \langle\langle x, m\rangle, k\rangle \wedge i < k \wedge \beta(x, m, i) = y), \\ \operatorname{len}(w) = k &\iff \exists x, m \leqslant w\, w = \langle\langle x, m\rangle, k\rangle. \end{aligned}$$

The meaning is that w encodes a sequence of length $\operatorname{len}(w)$ whose ith entry is $\operatorname{seq}(w, i)$ for $i < \operatorname{len}(w)$. (The functions as defined are partial—w only codes a sequence if it is of the form $\langle\langle x, m\rangle, k\rangle$ for some x, m, k. This will not be a concern.)

Theorem 3.16. *Every semidecidable set of natural numbers $X \subseteq \mathbb{N}$ is Σ_1-definable; i.e., there exists a Σ_1-formula $\sigma(x)$ such that*

$$n \in X \iff \mathbb{N} \models \sigma(\overline{n}) \qquad \textit{for all } n \in \mathbb{N}.$$

Proof. Fix a Turing machine $M = \langle Q, \Sigma, \Gamma, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta\rangle$ that semidecides X, or more precisely, the set of strings $\{w \in \{1, 2\}^* : \ulcorner w \urcorner \in X\}$ (see Definition 2.19). Thus, $\Sigma = \{1, 2\}$; we assume w.l.o.g. that the elements of Q and Γ are natural numbers as well (in particular, we identify $\sqcup$ with some natural number $\neq 1, 2$).

We represent configurations of M by natural numbers using sequence encoding: we cannot literally follow Definition 2.2 as we cannot encode infinite sequences, thus we represent a configuration as (a code of) a sequence $\langle q, h, w_0, \dots, w_s\rangle$ where $q \in Q$ is the current state, h is the head position, w_i is the content of ith cell of the tape, and $s \geqslant h$ is such that $w_i = \sqcup$ for all $i > s$. Note that the representation of a given configuration is non-unique, because the representation may use arbitrarily large s, and regardless of that, a given finite sequence can be coded by infinitely many different numbers.

Working with this representation, we will present formulas $\text{Initial}(u, x)$ expressing 'u is the initial configuration on input x', $\text{Accepting}(u)$ expressing 'u is an

accepting configuration', and $\mathrm{Yields}(u, v)$ expressing 'u yields v'. Then we can define X by the formula

$$\begin{aligned}\sigma(x) = \exists w \big[\mathrm{len}(w) \geqslant 1 \wedge \mathrm{Initial}\big(\mathrm{seq}(w, 0), x\big) \\ \wedge \mathrm{Accepting}\big(\mathrm{seq}(w, \mathrm{len}(w) - 1)\big) \\ \wedge \forall i < \mathrm{len}(w) - 1\, \mathrm{Yields}\big(\mathrm{seq}(w, i), \mathrm{seq}(w, i + 1)\big)\big]\end{aligned}$$

expressing Definition 2.3. Note that even though seq, len, and -1 are not L_{PA}-terms, they are Δ_0-functions, hence we can eliminate them from the formula while using only bounded quantifiers: e.g., an atomic formula of the form $\theta(\mathrm{seq}(w, i))$ can be rewritten as $\exists u \leqslant w\, (\theta(u) \wedge \mathrm{seq}(x, i) = u)$, where we can replace $\mathrm{seq}(x, i) = u$ with its Δ_0 definition. Thus, we can write $\sigma(x)$ as a Σ_1-formula as long as Initial and Accepting are Σ_1-formulas (whose initial existential quantifiers can be prenexed out of the square bracket), and Yields is a Δ_0-formula.

It remains to define the formulas Initial, Accepting, and Yields with the properties above. Again, we can use Δ_0-functions such as seq and len freely.

We can define

$$\begin{aligned}\mathrm{Accepting}(u) &\equiv \mathrm{seq}(u, 0) = q_{\mathrm{acc}}, \\ \mathrm{Yields}(u, v) &\equiv \bigvee_{\substack{\langle q,a\rangle \in Q\times\Gamma \\ \delta(q,a)=\langle q',a',t\rangle}} \mathrm{Next}_{q,a,q',a',t}(u, v),\end{aligned}$$

where $\mathrm{Next}_{q,a,q',a',t}(u, v)$ denotes

$$\begin{aligned}&\mathrm{seq}(u, 0) = q \\ &\wedge\, \mathrm{seq}(u, \mathrm{seq}(u, 1) + 2) = a \\ &\wedge\, \mathrm{len}(v) \geqslant \max\{\mathrm{len}(u), \mathrm{seq}(v, 1) + 3\} \\ &\wedge\, \forall i < \mathrm{len}(v)\ \mathrm{seq}(v, i) = \begin{cases} q' & i = 0, \\ \mathrm{seq}(u, 1) + 1 & i = 1\ \&\ t = R, \\ \max\{\mathrm{seq}(u, 1) - 1, 0\} & i = 1\ \&\ t = L, \\ a' & i = \mathrm{seq}(u, 1) + 2, \\ \mathrm{seq}(u, i) & 2 \leqslant i < \mathrm{len}(u)\ \& \\ & i \neq \mathrm{seq}(u, 1) + 2, \\ \sqcup & i \geqslant \mathrm{len}(u) \end{cases}\end{aligned}$$

(the last expression can be written using disjunctions and conjunctions as there are only a fixed number of cases).

The main problem with description of the initial configuration is to check that the digits $a_i \in \{1, 2\}$ on the tape form the bijective base-2 representation

$$x = \sum_{j<k} 2^j a_j$$

of the given input $x \in \mathbb{N}$. In order to do this, we use an auxiliary sequence w with values

$$\mathrm{seq}(w, i) = \sum_{j<k-i} 2^j a_{i+j}$$

for $i \leqslant k$, which satisfies the backwards recurrence

$$\mathrm{seq}(w, k) = 0, \qquad \mathrm{seq}(w, i) = 2\,\mathrm{seq}(w, i+1) + a_i,$$

and we check that $\mathrm{seq}(w, 0) = x$. Below, $a_i = \mathrm{seq}(u, i+2)$ and $k = \mathrm{len}(u) - 2 = \mathrm{len}(w) - 1$:

$$\begin{aligned}\mathrm{Initial}(u, x) \equiv\ & \mathrm{seq}(u, 0) = q_0 \wedge \mathrm{seq}(u, 1) = 0 \\ & \wedge \exists w \Big(\mathrm{len}(w) = \mathrm{len}(u) - 1 \\ & \qquad \wedge \mathrm{seq}(w, 0) = x \\ & \qquad \wedge \mathrm{seq}(w, \mathrm{len}(w) - 1) = 0 \\ & \qquad \wedge \forall i < \mathrm{len}(w) - 1) \\ & \qquad\qquad \mathrm{seq}(w, i) = 2\,\mathrm{seq}(w, i+1) + \mathrm{seq}(u, i+2) \Big)\end{aligned}$$

QED

3.4 Undecidability and incompleteness

Definition 3.17. If T is a theory in a finite language L, then T is said to be *decidable* if $\mathrm{Thm}(T) = \{\phi : T \vdash \phi\}$ is a decidable set.

A theory T in a finite language L is said to be *recursively axiomatized*, or *computably axiomatized*, if T, considered as a set of axioms (i.e., without the requirement of deductive closure), is decidable.

Definition 3.18. A theory T in the language of arithmetic is Σ_1*-sound* if all Σ_1-sentences σ provable in the theory are true; i.e.,

$$T \vdash \sigma \implies \mathbb{N} \models \sigma.$$

Observe that any Σ_1-sound theory is consistent.

We come to our first version of Gödel's first incompleteness theorem, though this is more properly called the *undecidability theorem*:

Theorem 3.19 (Kleene's undecidability theorem)**.**
Every Σ_1-sound theory $T \supseteq \mathsf{Q}$ is undecidable.

Proof. Let $X \subseteq \mathbb{N}$ be an undecidable but semidecidable set, which exists by Theorem 2.23 and Definition 2.19, and σ be a Σ_1-definition of X, which exists by Theorem 3.16. Then $n \mapsto \sigma(\overline{n})$ is a computable function that provides a many-one reduction of X to $\mathrm{Thm}(T)$, as

$$n \in X \iff \mathbb{N} \models \sigma(\overline{n}) \iff T \vdash \sigma(\overline{n}).$$

In the second equivalence, '$\Rightarrow$' follows from the Σ_1-completeness of $\mathsf{Q} \subseteq T$, and '$\Leftarrow$' from the Σ_1-soundness of T. Thus, $\mathrm{Thm}(T)$ is undecidable. QED

Exercise 3.20. The following sets and functions are computable for a fixed finite language L:

(1) The set of L-terms.

(2) The set of L-formulas.

(3) $\{(\phi, x) : x \text{ is a free variable of a formula } \phi\}$.

(4) The substitution function: given a formula ϕ, a variable x and a term t, compute $\phi(t/x)$.

(5) $\{(\Gamma, \phi, \pi) : \pi \text{ is a proof of } \phi \in \mathrm{Form}_L \text{ from a finite set } \Gamma \subseteq \mathrm{Form}_L\}$.

Lemma 3.21.

(1) *Every recursively axiomatized theory is semidecidable.*

(2) *Every complete, recursively axiomatized theory is decidable.*

Proof.

(1) Given ϕ, we exhaustively enumerate all pairs Γ, π. We accept if π is a proof of ϕ from Γ and all $\psi \in \Gamma$ are in T.

(2) $\mathrm{Thm}(T)$ is semidecidable by (1). Then

$$\Sigma^* \setminus \mathrm{Thm}(T) = \{\phi : \phi \text{ is not an } L\text{-sentence}\} \cup \{\phi : T \vdash \neg\phi\}$$

is also semidecidable and, therefore, $\mathrm{Thm}(T)$ is decidable. QED

Remark 3.22. Conversely, every semidecidable theory is recursively axiomatizable (i.e., equivalent to a recursively axiomatized theory). It might be also useful to mention that every consistent decidable theory has a complete consistent decidable extension.

We infer the proper statement of Gödel's first incompleteness theorem:

Theorem 3.23 (Gödel)**.** *If T is Σ_1-sound, recursively axiomatized extension of Q, then T is incomplete.*

Proof. If T were complete, then T would be decidable, contradicting Theorem 3.19. QED

Definition 3.24. The *Entscheidungsproblem* for a given finite language L is

$$\{\phi : \phi \text{ is an } L\text{-sentence}, \ \models \phi\} = \mathrm{Thm}(\varnothing)$$

(where $\varnothing$ is the L-theory with an empty set of non-logical axioms).

Theorem 3.25 (Church)**.** *The Entscheidungsproblem for L_{PA} is undecidable.*

Proof. Let α be the conjunction of axioms of Q. Then using the deduction lemma, $\mathrm{Thm}(\mathsf{Q}) \leqslant_m \mathrm{Thm}(\varnothing)$ via the reduction $\phi \mapsto (\alpha \to \phi)$:

$$\mathsf{Q} \vdash \phi \iff \vdash \alpha \to \phi.$$

Thus, the undecidability of Q implies the undecidability of $\mathrm{Thm}(\varnothing)$. QED

A few closing remarks:

(1) The assumption of Σ_1-soundness in Theorems 3.19 and 3.23 may be reduced to plain consistency; this is the Gödel–Rosser theorem. More explicitly, consistent extensions of Q are undecidable, and therefore incomplete if recursively axiomatized.

(2) The undecidability and incompleteness theorems hold for theories T that merely *interpret* Q rather than outright include it. Roughly speaking, this means that in the language of T, there are formulas defining a class of 'natural numbers' and the arithmetic operations from L_{PA} on this class in such a way that T proves the corresponding translations of all axioms of Q.

   For example, ZFC interprets Q (and even PA) by defining the standard model of arithmetic on ω. But much weaker theories of sets suffice (albeit the interpretation becomes more complicated):

(3) (Szmielew, Tarski) The *adjunctive set theory* (AST) with axioms

$$\exists z\, \forall t\, t \notin z,$$
$$\forall x\, \forall y\, \exists z\, \forall t\, (t \in z \leftrightarrow t \in x \vee t = y)$$

   interprets Q, thus consistent extensions of AST are undecidable, and incomplete if recursively axiomatized.

(4) The Entscheidungsproblem for L is undecidable iff L contains at least one at least binary symbol, or at least two unary functions.