If $S = \emptyset$, we just write

$$\vdash A$$

and call A a *formal theorem*. The same remarks as in section 1.11 apply here too. If we want to indicate the language L we are using, we explicitly write $S \vdash_L A$.

Before discussing examples of formal proofs, we have to explain the notion of substitutability. Recall that if A is a formula, x a variable and t a term, then $A[x/t]$ is the result of substituting every *free* occurrence of x in A by t. The question is: Is $\forall x A \Rightarrow A[x/t]$ logically valid?

Example 2.7.4 Consider

$$A : \exists y(x \neq y)$$
$$t : y.$$

Then $A[x/y]$ is the formula $\exists y(y \neq y)$, but

$$\forall x \exists y(x \neq y) \Rightarrow \exists y(y \neq y)$$

is clearly not valid.

The problem was that x was in the scope of $\exists y$ and we substituted x by y, which completely changed the meaning of this formula.

Definition 2.7.5 We say that t is *substitutable* for x in A if no *free* occurrence of x in A is within the scope of a quantifier $\forall z$ or $\exists z$, with z a variable occurring in t.

Examples 2.7.6

(i) x is substitutable for itself.

(ii) If t is a *closed* term, i.e., a term with no variables, then t is substitutable for any x.

(iii) If A has no quantifiers, *any* t is substitutable for x.

(iv) $f(x)$ is *not* substitutable for z in $\forall x(R(x, z) \Rightarrow \forall y Q(y))$.

(v) $h(z)$ is substitutable for x in $\forall y P(x, y)$.

So we restrict (d) only to the case t is substitutable for x in A. Then it is not hard to see that every formula of the form (d) is valid. Notice that, by example (i) above the formulas $\forall x A \Rightarrow A$ are in (d).

2.7.B Examples of Formal Proofs

Example 2.7.7 First notice that *if* $\vdash A$, *then* $\vdash \forall xA$ *for any variable* x (so $\vdash B$, where B is any generalization of A). To see this, let $A_1, \ldots, A_n = A$ be a formal proof of A. We claim that $\vdash \forall xA_i$, for $i = 1, \ldots, n$, so $\vdash \forall xA_n$, i.e., $\vdash \forall xA$. We can prove this by induction on $i = 1, \ldots, n$. If $i = 1$, then A_1 is a logical axiom and thus, by definition, so is $\forall xA_1$, so $\vdash \forall xA_1$. Assume this has been proved for all $j < i$ and consider A_i. If A_i is a logical axiom, we are done as in the case $i = 1$. Otherwise A_i comes by MP from A_j, A_k with $j, k < i$, i.e., A_k is of the form $A_j \Rightarrow A_i$. By induction hypothesis, $\vdash \forall xA_j$, $\vdash \forall x(A_j \Rightarrow A_i)$, so, since $\vdash \forall x(A_j \Rightarrow A_i) \Rightarrow (\forall xA_j \Rightarrow \forall xA_i)$ (by (b)), using MP twice, we see that $\vdash \forall xA_i$.

Example 2.7.8 If A is an instance of a tautology, then $\vdash A$, using the completeness theorem for propositional logic. So, using also example 2.7.7, we see that $\vdash \forall y_1 \ldots \forall y_n A$, for any $y_1, \ldots, y_n$ and any instance of a tautology A, i.e., any generalization of an instance of a tautology is a formal theorem, so we can use it freely in a formal proof.

Example 2.7.9 $\vdash \forall x(P(x) \Rightarrow \exists yP(y))$ or, recalling our abbreviations,

$$\vdash \forall x(P(x) \Rightarrow \neg\forall y\neg P(y)).$$

Here is why:

1. $\vdash \forall x[(\forall y\neg P(y) \Rightarrow \neg P(x)) \Rightarrow (P(x) \Rightarrow \neg\forall y\neg P(y))]$
   (This is a generalization of the instance of the tautology

   $$(A \Rightarrow \neg B) \Rightarrow (B \Rightarrow \neg A),$$

   with $A : \forall y\neg P(y),\ B : P(x)$.) For simplicity, abbreviate

   $$C : (\forall y\neg P(y) \Rightarrow \neg P(x))$$
   $$D : (P(x) \Rightarrow \neg\forall y\neg(P(y)))$$

2. $\vdash \forall x(C \Rightarrow D) \Rightarrow (\forall xC \Rightarrow \forall xD)$ (logical axiom (b))
3. $\vdash \forall xC \Rightarrow \forall xD$ (MP 1, 2)
4. $\vdash \forall x(\forall y\neg P(y) \Rightarrow \neg P(x))$ (logical axiom (d))
   (noticing that x is substitutable for y in $\neg P(y)$)
5. $\vdash \forall xD$ (MP 3, 4)
   (i.e., $\vdash \forall x(P(x) \Rightarrow \neg\forall y\neg P(y))$)

2.7.C Metatheorems

As in the case of propositional logic, we will prove a few metatheorems, which again formally correspond to common proof techniques.

Proposition 2.7.10 (Tautology Theorem) *If $S \vdash A_1, \ldots, S \vdash A_n$ and the set $\{A_1, \ldots, A_n\}$ tautologically implies B, i.e., $(A_1 \Rightarrow (A_2 \Rightarrow \ldots \Rightarrow (A_n \Rightarrow B)\ldots))$ is an instance of a tautology, then $S \vdash B$.*

Proof. We have $\vdash (A_1 \Rightarrow \ldots \Rightarrow (A_n \Rightarrow B)\ldots))$, so, since $S \vdash A_1, \ldots, S \vdash A_n$, by applying MP several times we get $S \vdash B$. ⊣

Proposition 2.7.11 (Deduction Theorem)

$$S \cup \{A\} \vdash B \text{ iff } S \vdash (A \Rightarrow B)$$

Proof. Exactly as in propositional logic. ⊣

Proposition 2.7.12 (Proof by Contradiction) *If $S \cup \{A\}$ is formally inconsistent, then $S \vdash \neg A$.*

Proof. Exactly as in propositional logic. ⊣

Example 2.7.13 $\vdash \exists x\,(x = x)$. We leave the verification as an exercise. It may be easier by using also some of the metatheorems that follow.

Proposition 2.7.14 (Proof by Contrapositive) *If $S \cup \{A\} \vdash \neg B$, then $S \cup \{B\} \vdash \neg A$.*

Proof. Exactly as in propositional logic. ⊣

Proposition 2.7.15 (Generalization Theorem) *If $S \vdash A$ and x is not free in any formula in S, then $S \vdash \forall x A$.*

Proof. By induction on proofs of A from S.

Basis.

(i) A is a logical axiom. Then $\forall x A$ is also a logical axiom, so clearly $S \vdash \forall x A$.

(ii) A is in S. Then, by hypothesis, x is not free in A. So $A \Rightarrow \forall x A$ is a logical axiom. By MP then, $S \vdash \forall x A$.

Induction Step. Assume that $S \vdash \forall x A$, $S \vdash \forall x(A \Rightarrow B)$ in order to show that $S \vdash \forall x B$. Since $\forall x(A \Rightarrow B) \Rightarrow (\forall x A \Rightarrow \forall x B)$ is a logical axiom, this follows by MP applied twice. $\dashv$

Remark. The assumption that x is not free in any formula in S is necessary, as the following example shows:

$$P(x) \vdash P(x)$$

but

$$P(x) \nvdash \forall x P(x).$$

(Otherwise, by the easy half of Gödel's Theorem—the soundedness property $S \vdash A$ implies $S \models A$,—we would have $P(x) \models \forall x P(x)$, which is easily false.)

Example 2.7.16 $\forall x \forall y A \vdash \forall y \forall x A$.

Proof. By Generalization, it is enough to show

$$\forall x \forall y A \vdash \forall x A,$$

and by Generalization once again, it is enough to show

$$\forall x \forall y A \vdash A.$$

Now

$$\vdash \forall x \forall y A \Rightarrow \forall y A$$

(logical axiom (d)) and

$$\vdash \forall y A \Rightarrow A$$

(logical axiom (d)), so by MP twice,

$$\forall x \forall y \vdash A.$$

Example 2.7.17 $\vdash \exists x \forall y A \Rightarrow \forall y \exists x A$.

Proof. By the Deduction Theorem, it is enough to show that

$$\exists x \forall y A \vdash \forall y \exists x A$$

or, by Generalization,

$$\exists x \forall y A \vdash \exists x A$$

or, recalling our abbreviations,

$$\neg \forall x \neg \forall y A \vdash \neg \forall x \neg A$$

or, by Proof by Contrapositive and the Tautology Theorem,

$$\forall x \neg A \vdash \forall x \neg \forall y A.$$

Again by Generalization enough to prove

$$\forall x \neg A \vdash \neg \forall y A$$

or by Proof by Contradiction enough to prove that

$$S = \{\forall x \neg A,\ \forall y A\}$$

is formally inconsistent.

But $\forall x \neg A \Rightarrow \neg A,\ \forall y A \Rightarrow A$ are both logical axioms (d), so by MP

$$S \vdash A,\ S \vdash \neg A$$

and we are done.

Example 2.7.18 $\vdash (A \Rightarrow \forall x B) \Leftrightarrow \forall x (A \Rightarrow B)$, provided x is not free in A.

Proof. It is enough to show (by the Tautology Theorem):

$$\vdash (A \Rightarrow \forall x B) \Rightarrow \forall x (A \Rightarrow B) \qquad (*)$$
$$\vdash \forall x (A \Rightarrow B) \Rightarrow (A \Rightarrow \forall x B) \qquad (**)$$

$(*)$: By the Deduction Theorem enough to show that

$$A \Rightarrow \forall x B \vdash \forall x (A \Rightarrow B),$$

so by Generalization enough to show that

$$A \Rightarrow \forall x B \vdash A \Rightarrow B,$$

or by the Deduction Theorem

$$A, A \Rightarrow \forall x B \vdash B$$

which is clear by MP and the fact that $\forall x B \Rightarrow B$ is a logical axiom (d).

($**$): By the Deduction Theorem, it is enough to show that

$$\forall x(A \Rightarrow B),\ A \vdash \forall x B$$

But $\forall x(A \Rightarrow B) \Rightarrow (\forall x A \Rightarrow \forall x B)$ is a logical axiom (d) and so is $A \Rightarrow \forall x A$, so this is clear by MP.

Proposition 2.7.19 (Generalization on Constants) *Let L be a first-order language, S a set of formulas in L, $A(x)$ a formula in L and c a constant symbol not in L. Then if $S \vdash_{L\cup\{c\}} A[x/c]$, we have $S \vdash_L \forall x A$.*

Proof. Since $S \vdash A[x/c]$, let $S_0 \subseteq S$ be a finite subset of S such that $S_0 \vdash A[x/c]$. Say

$$A_1, A_2, \ldots, A_k = A[x/c]$$

be a proof of $A[x/c]$ from S_0 (in $L\cup\{c\}$). Let $A'_1, \ldots, A'_k$ result by substituting c by y in $A_1, \ldots, A_k$, where y is a variable not occurring in $S_0, A_1, \ldots, A_k, A$. Then it can be easily checked by induction that $A'_1, \ldots, A'_k = A[x/y]$ is a proof from S_0 (in L). So

$$S_0 \vdash A[x/y]$$

Since y does not occur in S_0, we have by Generalization that

$$S_0 \vdash \forall y A[x/y].$$

But clearly x is substitutable for y in $A[x/y]$ and

$$A[x/y][y/x] = A,$$

so

$$\forall y A[x/y] \Rightarrow A$$

is a logical axiom, thus

$$\forall y A[x/y] \vdash A$$

(by MP), so by Generalization again

$$\forall y A[x/y] \vdash \forall x A,$$

and by MP

$$S_0 \vdash \forall x A,$$

so

$$S \vdash \forall x A.$$

⊣

Example 2.7.20 $\forall x \forall y R(x, y) \vdash \forall y \forall x R(x, y)$ (a different argument).

Proof. Enough to show

$$\forall x \forall y R(x, y) \vdash \forall x R(x, d),$$

where d is a constant symbol, or again

$$\forall x \forall y R(x, y) \vdash R(c, d),$$

where c is a constant symbol different from d.

But $\forall x \forall y R(x, y) \Rightarrow \forall y R(c, y)$ is a logical axiom (d) and $\forall y R(c, y) \Rightarrow R(c, d)$ is a logical axiom (d), so by MP twice, $\forall x \forall y R(x, y) \vdash R(c, d)$.

2.8 The Gödel Completeness Theorem

Kurt Gödel was an Austrian logician, widely considered one of the most important and influential logicians of all time. He was born in 1906 in Brno, then part of the Austro-Hungarian empire, and now a Czech city. Gödel died in Princeton in 1978.

Gödel's work in mathematical logic produced three celebrated results: First, the *completeness theorem*, whose proof we present in this section: First-order logic is complete in the sense that a theory proves a formula if and only if all models of the theory are also models of the formula. Second, the *incompleteness theorems*, published in 1931, one year after finishing his doctorate at the University of Vienna. The incompleteness results state that if a theory is "understandable enough" (meaning that its axioms can be generated algorithmically) and powerful enough to prove a small fragment of Peano arithmetic, then either it is inconsistent or else there are true (arithmetic) sentences that it cannot prove. In fact, if it proves enough of Peano arithmetic, then statements about certain formulas being provable

from the theory can be represented by statement about numbers, and using this representation one can write a sentence Con that states that the theory is consistent. Then, either the theory is inconsistent, or else Con itself is not provable in the theory. This result destroyed a program initiated by Hilbert trying to show, by self-contained mathematical methods, the consistency of mathematics. Gödel's third main result is in set theory, where he introduced the universe L of *constructible sets* and proved that L is a model of set theory, together with the axiom of choice and the generalized continuum hypothesis.

Gödel also made contributions to intuitionistic logic and relativity theory. In 1933, he met Einstein while travelling in the US. The two of them became good friends and after his move to Princeton, Einstein was one of the very few people with whom Gödel kept personal contact.

He moved to the States in 1940 because his former association with Jewish members of the so called Vienna circle (an influential group of mathematicians and philosophers of which Alfred Tarski, the logician who introduced the notion of truth ($\models$), was also a member) made it difficult for him to keep his position in Vienna under the Nazi regime.

Gödel suffered from delusions that eventually developed into an acute case of paranoia. He believed conspirators tried to assassinate him with poison gas and eventually thought that they were trying to poison his food. In 1977, due to illness, his wife could no longer help him with his meals, and he refused to eat any food at all, starving himself to death.

Theorem 2.8.1 (Gödel) *Let L be a first-order language and S a set of formulas in L, A a formula in L. Then*

$$S \models A \textit{ iff } S \vdash A.$$

Proof. The direction

$$S \vdash A \text{ implies } S \models A$$

is easy and is called the *Soundedness Property* of the proof system. It follows immediately from the fact that every logical axiom is logically valid and that MP preserves validity.

We will now prove the *Completeness Property* of the proof system, i.e.,

$$S \models A \text{ implies } S \vdash A.$$

As is the case of propositional logic (section 1.11.C), it is enough to show the following:

If a set of formulas S is formally consistent, then it is satisfiable. $(*)$

(As before, we call S *formally inconsistent* if for some formula A we have $S \vdash A$ and $S \vdash \neg A$, and otherwise we call S *formally consistent*.)

For simplicity, we will give in detail the proof of $(*)$ in case S is a set of *sentences*. (The case of arbitrary formulas follows from this by a little extra argument.) So we have a formally consistent set of sentences S and we have to find a model of S. First, we need some terminology:

Definition 2.8.2 A *formal theory* is a set of sentences T closed under formal provability, i.e. $T \vdash A$ (A a sentence) implies $A \in T$.

Definition 2.8.3 A set of sentences T is *complete* if for any sentence A we have either $A \in T$ or $\neg A \in T$.

It is then clear that if T is formally consistent and complete, then for any sentence A, exactly one of A and $\neg A$ belongs to T.

Definition 2.8.4 A *Henkin theory* is a formal theory which has the following extra property:

"For any formula $A(x)$ there is a constant symbol $c = c_A$ such that
$\neg \forall x A(x) \Rightarrow \neg A[x/c]$ is in T."

We call c_A a *Henkin witness* for $\neg \forall x A(x)$.

We prove two lemmas from which $(*)$ follows immediately.

Lemma 2.8.5 *Suppose L' is a first-order language, T' a formally consistent, complete Henkin theory in L'. Then T' has a model $\mathcal{M}$.*

Lemma 2.8.6 *Let L be a first-order language and S a formally consistent set of sentences in L. Then there is a first-order language $L' \supseteq L$ obtained by adding some constant symbols to L, and a formally consistent, complete, Henkin theory T' in L' such that $T' \supseteq S$.*

Proof of of Lemma 2.8.5. Let

$$A = \{t : t \text{ is a closed term in } L'\}.$$

Define an equivalence relation $\sim$ on A as follows:

$$s \sim t \text{ iff } s = t \in T'$$

($s = t$ is a sentence in L). We have first to check that this is indeed an equivalence relation:

(i) $s \sim s$.

This means that $s = s \in T'$. But $\forall x(x = x) \Rightarrow s = s$ is a logical axiom (d) and $\forall x(x = x)$ is a logical axiom (e), so, by MP, $\vdash s = s$, so $T' \vdash s = s$, thus $s = s \in T'$.

(ii) $s \sim t$ implies $t \sim s$.

We have that $s = t \in T'$. Now $\forall x \forall y(x = y \Rightarrow y = x)$ is a logical axiom (e) and $\vdash \forall x \forall y(x = y \Rightarrow y = x) \Rightarrow (s = t \Rightarrow t = s)$, so by MP, $\vdash (s = t) \Rightarrow (t = s)$ so as $s = t \in T'$, $T' \vdash t = s$, thus $t = s \in T'$ (since T' is a formal theory), i.e., $t \sim s$.

(iii) $s \sim t$ and $t \sim u$ imply $s \sim u$.

Again $\forall x \forall y \forall z((x = y \wedge y = z) \Rightarrow x = z)$ is a logical axiom (e) and $\vdash \forall x \forall y \forall x((x = y \wedge y = z) \Rightarrow x = z) \Rightarrow ((s = t \wedge t = u) \Rightarrow s = u)$ by using three times logical axiom (d) and MP, so , by MP, $\vdash (s = t \wedge t = u) \Rightarrow s = u$, so $T \vdash (s = t \wedge t = u) \Rightarrow s = u$ and since $s = t$, $t = u \in T'$ it follows that $T' \vdash s = u$, so $s = u \in T'$.

Denote by

$$[s] = \{t : t \sim s\}$$

the equivalence class of $s \in A$, and by

$$M = \{[s] : s \text{ a closed term}\}$$

the set of all equivalence classes (the quotient set of A modulo $\sim$). This will be the underlying universe of the model of T'. We will now define the interpretations of the relation and function symbols of L':

Let f be an n-ary function symbol. Define its interpretation $f^{\mathcal{M}}$ by

$$f^{\mathcal{M}}([s_1], \ldots, [s_n]) = [f(s_1, \ldots, s_n)].$$

One has to show that this is well-defined, i.e., if $s_1 \sim t_1, \ldots, s_n \sim t_n$, then

$$f(s_1, \ldots, s_n) \sim f(t_1, \ldots, t_n).$$

This follows easily from the fact that

$$\vdash (s_1 = t_1 \wedge \cdots \wedge s_n = t_n) \Rightarrow f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n),$$

which can be proved by arguments similar to those before (using logical axioms (e)).

As a special case, if c is a constant symbol we have

$$c^{\mathcal{M}} = [c].$$

For further reference, note also that for any closed term s we have

$$s^{\mathcal{M}} = [s].$$

(This can be easily proved by induction on the construction of s.)

Now let R be an m-ary relation symbol. Define its interpretation $R^{\mathcal{M}}$ by

$$R^{\mathcal{M}}([s_1], \ldots, [s_m]) \text{ iff } R(s_1, \ldots, s_m) \in T'.$$

Again we have to show that this is well-defined, i.e. that $s_1 \sim t_1, \ldots, s_m \sim t_m$ imply that

$$R(s_1, \ldots, s_m) \in T' \text{ iff } R(t_1, \ldots, t_m) \in T'.$$

This follows from the fact that

$$\vdash (s_1 = t_m \wedge \cdots \wedge s_m = t_m) \Rightarrow [R(s_1, \ldots, s_m) \Leftrightarrow R(t_1, \ldots, t_m)].$$

We have now completed the description of the structure $\mathcal{M}$. It remains to show that

$$\mathcal{M} \models T'.$$

Actually we will show that for each sentence A of L' we have

$$\mathcal{M} \models A \text{ iff } A \in T'. \qquad (**)$$

For each sentence A, let

$$N(A) = \text{ the total number of } \neg, \Rightarrow, \forall \text{ in } A.$$

We will prove $(**)$ by induction on $N(A)$.

Basis. $N(A) = 0$. Then A is atomic, i.e., $s = t$ or $R(s_1, \ldots, s_m)$ for closed terms $s, t, s_1, \ldots, s_m$. For the case of $s = t$ we have

$$\begin{aligned} \mathcal{M} \models s = t \text{ iff } s^{\mathcal{M}} = t^{\mathcal{M}} &\text{ iff } [s] = [t] \\ &\text{iff } s \sim t \text{ iff } s = t \in T'. \end{aligned}$$

For the case of $R(s_1, \ldots, s_m)$ we have

$$\begin{aligned} \mathcal{M} \models R(s_1, \ldots, s_m) &\text{ iff } R^{\mathcal{M}}(s_1^{\mathcal{M}}, \ldots, s_m^{\mathcal{M}}) \\ &\text{iff } R^{\mathcal{M}}([s_1], \ldots, [s_m]) \\ &\text{iff } R(s_1, \ldots, s_m) \in T'. \end{aligned}$$

Induction step. Assume $(**)$ has been proved for all sentences A' with $N(A') \leq n$. Say A is such that $N(A) = n + 1$.

Case 1. A is of the form $\neg B$. Then $N(B) = n$, so by induction hypothesis

$$\mathcal{M} \models B \text{ iff } B \in T',$$

thus

$$\begin{aligned} \mathcal{M} \models A \text{ iff } M \not\models B &\text{ iff } B \notin T' \\ &\text{iff } \neg B \in T' \text{ (as } T' \text{ is formally consistent and complete)} \\ &\text{iff } A \in T'. \end{aligned}$$

Case 2. A is of the form $B \Rightarrow C$. Then $N(B), N(C) \leq n$, so by induction hypothesis

$$\begin{aligned} \mathcal{M} \models B \text{ iff } B \in T' \\ \mathcal{M} \models C \text{ iff } C \in T'. \end{aligned}$$

Thus we have

$$\begin{aligned} \mathcal{M} \models A \text{ iff } \mathcal{M} \not\models B \text{ or } \mathcal{M} \models C \\ \text{iff } B \notin T' \text{ or } C \in T' \\ \text{iff } \neg B \in T' \text{ or } C \in T' \\ \text{iff } B \Rightarrow C \in T' \qquad (+) \\ \text{iff } A \in T' \end{aligned}$$

To see (+), notice that if $\neg B \in T'$, then since $\neg B \Rightarrow (B \Rightarrow C)$ is an instance of a tautology, so $\vdash \neg B \Rightarrow (B \Rightarrow C)$, we have $T' \vdash B \Rightarrow C$, so $B \Rightarrow C \in T'$. Similarly if $C \in T'$, since $C \Rightarrow (B \Rightarrow C)$ is a logical axiom (a). Conversely, if $B \Rightarrow C \in T'$ but $\neg B \notin T'$ and $C \notin T'$, then $B \in T'$ and $\neg C \in T'$, so by MP $C \in T'$ and $\neg C \in T'$, which is impossible, as T' is formally consistent.

Case 3. A is of the form $\forall x B(x)$. Then $N(B) = n$, so also $N(B[x/t]) = n$ for any closed term t, thus by induction hypothesis:

$$\mathcal{M} \models B[x/t] \text{ iff } B[x/t] \in T'.$$

We will show that

$$\mathcal{M} \models \forall x B(x) \text{ iff } \forall x B(x) \in T'.$$

If $\mathcal{M} \models \forall x B(x)$, then for any closed term t, we have

$$\mathcal{M} \models B[x/t]$$

(notice that $\forall x B(x) \Rightarrow B[x/t]$ is logically valid). So $B[x/t] \in T'$, for all closed terms t. If now $\forall x B(x) \notin T$, towards a contradiction, then $\neg \forall x B(x) \in T'$, so, as T' is a Henkin theory, for some constant symbol c we have $\neg B[x/c] \in T'$, a contradiction.

Conversely, assume $\forall x B(x) \in T'$. Then as $\forall x B(x) \Rightarrow B[x/t]$ is a logical axiom (d) for every closed term t, we have $B[x/t] \in T'$, so by induction hypothesis $\mathcal{M} \models B[x/t]$, i.e., $\mathcal{M} \models B[x \mapsto t^{\mathcal{M}}]$ or $\mathcal{M} \models B[x \mapsto [t]]$. But every element of $\mathcal{M}$ has the form $[t]$ for some such t, so $\mathcal{M} \models B[x \mapsto a]$ for every $a \in M$, i.e., $\mathcal{M} \models \forall x B(x)$. Lemma 2.8.5 ⊣

Proof of Lemma 2.8.6. Again for simplicity in exposition, we will assume that L is countable, i.e. all non-logical symbols in L can be enumerated in a sequence. It follows that one can enumerate in a sequence all formulas of L and so all sentences of L.

The proof of Lemma 2.8.6 will be done in two steps.

1st Step. We will first define a first-order language $L' \supseteq L$ obtained by adding to L a countable set of constant symbols, and a formally consistent Henkin theory S' in L' such that $S' \supseteq S$. Notice that L' is still countable.

2nd Step. We will find a formally consistent and complete theory $T' \supseteq S'$ in the language L'. Then notice that T' is also a Henkin theory (by definition), so we are done.

The second step can be done exactly as in the case of propositional logic (see the proof of sublemma 1.11.11), so we won't repeat it.

Thus it is enough to do the 1st step: Define recursively on n, a first order language L_n and a set of sentences S_n of L_n such that

$$L = L_0 \subseteq L_1 \subseteq L_2 \subseteq \dots$$

$$S = S_0 \subseteq S_1 \subseteq S_2 \subseteq \dots$$

as follows:

$$L_0 = L,\ S_0 = S$$

Assume L_n, S_n are given. Define then L_{n+1} by adding to L_n one *new* constant symbol (not in L_n and distinct from each other)

$$c = c_{\neg\forall x A(x)}$$

for each sentence of L_n of the form $\neg\forall x A(x)$. Finally obtain S_{n+1} by adding to S_n all the sentences

$$\neg\forall x A(x) \Rightarrow \neg A[x/c],$$

where $A(x)$ is a formula of L_n and c is the constant symbol associated to $\neg\forall x A(x)$.

Put now

$$L' = \bigcup_n L_n,\ S'' = \bigcup_n S_n.$$

Clearly $L' \supseteq L$ is obtained from L by adding to L countably many constant symbols. Let S' be the formal theory generated by S'', i.e.,

$$S' = \{A \text{ a sentence of } L' : S'' \vdash A\}.$$

Clearly $S \subseteq S'' \subseteq S'$, so it is enough to show that S' is a formally consistent Henkin theory.

That it is a Henkin theory is easy: If $A(x)$ is a formula of L', then $A(x)$ is a formula of some L_n, so if $c = c_{\neg\forall x A(x)}$, then c is a constant symbol in L_{n+1}, thus in L', and

$$\neg\forall x A(x) \Rightarrow \neg A[x/c]$$

is in S_{n+1}, thus in S'.

It remains to show that S' is formally consistent or equivalently that S'' is formally consistent, and for that it is enough to show each S_n is formally

consistent. This is done by induction on n. For $n = 0$, it is given as $S_0 = S$. So assume now that S_n is formally consistent. If S_{n+1} is formally inconsistent, towards a contradiction, there are formulas $A_1(y_1), \ldots, A_k(y_k)$ of L_n and new (i.e., not in L_n) distinct constants of L_{n+1}, say $c_1, \ldots, c_k$, such that

$$S_n \cup \{\neg\forall y_i A_i(y_i) \Rightarrow \neg A_i(c_i) : 1 \leq i \leq k\}$$

is formally inconsistent, where $A_i(c_i)$ abbreviates $A_i[y_i/c_i]$. Then by proof by contradiction

$$\underbrace{S_n \cup \{\neg\forall y_i A_i(y_i) \Rightarrow \neg A_i(c_i) : 1 \leq i \leq k-1\}}_{Q} \vdash \neg(\neg\forall y_k A_k(y_k) \Rightarrow \neg A_k(c_k)),$$

so, by using tautologies,

$$Q \vdash \neg\forall y_k A_k(y_k)$$

$$Q \vdash A_k(c_k).$$

But by Generalization on Constants, since c_k does not occur in the formulas in Q, we have that

$$Q \vdash \forall y_k A_k(y_k),$$

so Q is formally inconsistent. Proceeding successively in a similar fashion we eliminate one-by-one all $\neg\forall y_i A_i(y_i) \Rightarrow \neg A_i(c_i)$ for $i = k, \ldots, 1$, so we get that S_n is formally inconsistent, a contradiction. Lemma 2.8.6 ⊣

2.9 The Compactness Theorem

As an immediate corollary of the Gödel Completeness Theorem for First-Order Logic, we obtain the compactness theorem for first-order logic:

Theorem 2.9.1 (The Compactness Theorem, I) *Let L be a first-order language, S a set of sentences in L and A a sentence in L. Then if $S \models A$, there is a finite subset $S_0 \subseteq S$ such that $S_0 \models A$.*

Theorem 2.9.2 (The Compactness Theorem, II) *Let L be a first-order language and S a set of sentences of L. If every finite subset $S_0 \subseteq S$ has a model, S has a model.*

We will discuss some applications of this Compactness Theorem.

2.9.A Finiteness and Infinity

Theorem 2.9.3 *Let S be a set of sentences in L. If S has arbitrarily large finite models, then S has an infinite model.*

Proof. Let λ_n be the sentence

$$\lambda_n : \exists x_1 \exists x_2 \ldots \exists x_n (\bigwedge_{i<j} x_i \neq x_j).$$

Then for any structure $\mathcal{M} = \langle M, - \rangle$,

$$\mathcal{M} \models \lambda_n \text{ iff } \mathrm{card}(M) \geq n.$$

Let $T = S \cup \{\lambda_1, \lambda_2, \ldots\}$. Our hypothesis implies that every finite subset of T has a model. So by the Compactness Theorem, T has a model $\mathcal{M}$. Then $\mathcal{M} \models S$ and $\mathcal{M} \models \lambda_n$ for each n, so $\mathcal{M}$ is infinite (i.e., M is infinite). $\dashv$

Corollary 2.9.4 *"Finiteness" cannot be expressed in first-order logic, i.e., for any first-order language L there is no set of sentences S in L such that for any structure $\mathcal{M}$ of L*

$$\mathcal{M} \models S \quad \textit{iff } \mathcal{M} \quad \textit{is finite.}$$

If $\Lambda_\infty = \{\lambda_1, \lambda_2, \ldots\}$, then clearly

$$\mathcal{M} \models \Lambda_\infty \text{ iff } \mathcal{M} \text{ is infinite,}$$

so "infinity" can be characterized by an (infinite) set of sentences in first-order logic, namely Λ_∞. Can it be characterized by a finite set of sentences, or equivalently by a single sentence A? If it could, then we would have

$$\mathcal{M} \models A \text{ iff } \mathcal{M} \text{ is infinite,}$$

so

$$\mathcal{M} \models \neg A \text{ iff } \mathcal{M} \text{ is finite,}$$

contradicting the previous corollary. It follows that if

$$T = \mathrm{Con}(\Lambda_\infty),$$

then T is *not* finitely axiomatizable.

Consider the language $L = \{<\}$. Can we find a set of sentences S in L such that

$$\mathcal{M} = \langle M, <^{\mathcal{M}} \rangle \models S \text{ iff } \mathcal{M} \text{ is a well ordering?}$$

We can see that the answer is negative as follows: Assume such an S existed, towards a contradiction. Consider the language $L' = \{<, c_0, c_1, c_2, \dots\}$ obtained by adding to L an infinite list of constant symbols $c_0, c_1, \dots$ Let

$$T = S \cup \{c_1 < c_0, c_2 < c_1, c_3 < c_2, \dots\}.$$

Then we claim that any finite subset T_0 of T has a model. This is because T_0 is contained in $S \cup \{c_1 < c_0, c_2 < c_1, \dots, c_{n_0+1} < c_{n_0}\}$ for some large enough n_0. A model of T_0 is then

$$\mathcal{M}_0 = \langle \mathbb{N}, <, c_0^{\mathcal{M}_0}, c_1^{\mathcal{M}_0}, \dots \rangle,$$

where $c_i^{\mathcal{M}} = n_0 + 1 - i$, if $i \leq n_0 + 1$ and $c_i^{\mathcal{M}} = 0$ if $i > n_0 + 1$. So, by the Compactness Theorem, T has a model

$$\mathcal{M} = \langle M, <^{\mathcal{M}}, c_0^{\mathcal{M}}, c_1^{\mathcal{M}}, \dots \rangle.$$

Then let

$$A = \{c_0^{\mathcal{M}}, c_1^{\mathcal{M}}, \dots\} \subseteq M.$$

Since $c_0^{\mathcal{M}} >^{\mathcal{M}} c_1^{\mathcal{M}} >^{\mathcal{M}} c_2^{\mathcal{M}} >^{\mathcal{M}} \dots$, clearly A has no least element (in $<^{\mathcal{M}}$), so it is not a wellordering, contradicting that

$$\langle M, <^{\mathcal{M}} \rangle \models S.$$

2.9.B Non-standard Models of Arithmetic

There are two main types of theories we have discussed. Many, such as the theories of groups, partial orders, ordered fields, and so on, are intended to have many models. The models of the theory of groups are all the objects which we call groups, and so on. From the standpoint of the first-order theory, all we can say about an arbitrary group is that it satisfies certain axioms and everything that follows from them. Of course, there are many statements independent of the axioms, and these may be true or false of individual groups. This is what makes group theory interesting.

The other type of theories are those intended (at least originally) to represent a single model. For example, the theory of arithmetic is constructed

with the structure of the natural numbers $\mathcal{N}$ in mind. A very natural question to ask is, "Do these theories also have multiple models?" For example, are there other structures which "act like $\mathcal{N}$" as far as the first-order theory can tell, but are not in fact the same? The answer is yes, and the consequences are interesting and far-reaching. We start by defining what we are looking for:

Definition 2.9.5 A *model of (complete) arithmetic* is a structure of L_{ar},

$$\mathcal{M} = \langle M, 0^{\mathcal{M}}, S^{\mathcal{M}}, +^{\mathcal{M}}, \cdot^{\mathcal{M}}, <^{\mathcal{M}} \rangle$$

which is a model of $\mathrm{Th}(\mathcal{N})$, i.e. for any sentence A in the language of arithmetic

$$\mathcal{M} \models A \text{ iff } \mathcal{N} \models A.$$

What does a general model of arithmetic look like? Well, to start with, it must contain objects $n^{\mathcal{M}}$ which "look like" the natural numbers $n \in \mathbb{N}$. For any such $n \in \mathbb{N}$, let

$$n^{\mathcal{M}} = \underline{n}^{\mathcal{M}},$$

where $\underline{n}$ is the term in L_{ar} given by:

$$\underline{n} = \underbrace{S(S(S \dots S}_{n \text{ times}}(0) \dots))$$

(so that $\underline{n}^{\mathcal{N}} = n$). Let

$$\mathbb{N}^{\mathcal{M}} = \{0^{\mathcal{M}}, 1^{\mathcal{M}}, \dots\}.$$

Then it is not hard to see that $\mathbb{N}^{\mathcal{M}}$ is closed under $S^{\mathcal{M}}, +^{\mathcal{M}}, \cdot^{\mathcal{M}}$, so it defines a structure

$$\mathcal{N}^{\mathcal{M}} = \langle \mathbb{N}^{\mathcal{M}}, 0^{\mathcal{M}}, S^{\mathcal{M}}|\mathbb{N}^{\mathcal{M}}, +^{\mathcal{M}}|\mathbb{N}^{\mathcal{M}}, \cdot^{\mathcal{M}}|\mathbb{N}^{\mathcal{M}}, <^{\mathcal{M}}|\mathbb{N}^{\mathcal{M}} \rangle,$$

and the map $n \mapsto n^{\mathcal{M}}$ is an isomorphism of $\mathcal{N}$ with $\mathcal{N}^{\mathcal{M}}$. So we can simply identify $\mathcal{N}$ with $\mathcal{N}^{\mathcal{M}}$. We call the elements $n = n^{\mathcal{M}}$ the *standard elements* of $\mathcal{M}$. So we see that any model of arithmetic must contain, at least, an embedded "standard copy" of $\mathcal{N}$.

Now what if $\mathcal{M}$ has non-standard elements? What would they look like? They have to be "infinite": that is, we must have

$$n <^{\mathcal{M}} a$$

for all standard n. This is because

$$\mathcal{N} \models \forall x(x < \underline{n} \Rightarrow x = 0 \vee x = \underline{1} \vee \cdots \vee x = \underline{n-1}),$$

so $\mathcal{M} \models \forall x(x < \underline{n} \Rightarrow x = 0 \vee \cdots \vee x = \underline{n-1})$. So if $a \in \mathcal{M}$ is different than each n, then, since $<^{\mathcal{M}}$ is a linear order, if $n <^{\mathcal{M}} a$ fails, then we must have $a <^{\mathcal{M}} n$ (since $a \neq n^{\mathcal{M}}$), so $a = 0$, or $a = 1, \ldots,$ or $a = n-1$, a contradiction. So we have the picture

Do non-standard models of arithmetic exist? That is, models of arithmetic with non-standard elements (equivalently not isomorphic to $\mathcal{N}$)? The answer is yes, as we can see using the Compactness Theorem.

Consider the language

$$L' = \{0, S, +, \cdot, <\} \cup \{c\},$$

where c is a new constant symbol. Let S be the following set of sentences in L':

$$S = \mathrm{Th}(\mathcal{N}) \cup \{\underline{n} < c : n \in \mathbb{N}\}.$$

Then every finite subset S_0 of S has a model. This is because $S_0 \subseteq Th(\mathcal{N}) \cup \{\underline{n} < c : n \leq n_0\}$ for some large enough n_0. Then a model of S_0 is

$$\mathcal{N}_0 = \langle \mathbb{N}, 0, S, +, \cdot, <, c^{\mathcal{N}_0} \rangle$$

where $c^{\mathcal{N}_0} = n_0 + 1$. So, by the Compactness Theorem, S has a model

$$\mathcal{M} = \langle M, 0^{\mathcal{M}}, S^{\mathcal{M}}, +^{\mathcal{M}}, \cdot^{\mathcal{M}}, <^{\mathcal{M}}, c^{\mathcal{M}} \rangle.$$

Then if $a = c^{\mathcal{M}}$ we have

$$\underline{n}^{\mathcal{M}} = n <^{\mathcal{M}} a$$

for all $n \in \mathbb{N}$, i.e. a is non-standard.

Non-standard models such as this one can often be used in many areas of mathematics to simplify proofs and elucidate concepts more clearly. For example there are non-standard models of $\mathcal{R}$ that contain "infinitesimals"—non-standard elements which are positive yet smaller (under $<^{\mathcal{M}}$) than any

(embedded standard copy of a) positive real number—and they can be used to reformulate calculus without using limits. This field of study is called *infinitesimal analysis* or *non-standard analysis*.

Here are some basic examples. Let $A \subseteq \mathbb{R}$ and let $f : A \to \mathbb{R}$. Consider the structure

$$\mathcal{R} = (\mathbb{R}, \mathbb{N}, 0, 1, <, +, -, \times, /, f)$$

where $\mathbb{N}$, $/$ and f are seen as relations: $\mathbb{N}$ is seen as a unary relation, so $n \in \mathbb{N}$ iff $\mathbb{N}(n)$, $/$ is seen as a ternary relation, so $a/b = c$ for $b \neq 0$ iff $/(a, b, c)$ and f is seen as a binary relation, so $f(a) = b$ iff $f(a, b)$, this is simply so we can talk about f even if its domain is not all of $\mathbb{R}$, and similarly for $/$.

A *nonstandard model* of analysis is a structure

$$^*\mathcal{R} = (^*\mathbb{R}, ^*\mathbb{N}, ^*0, ^*1, ^*<, ^*+, ^*-, ^*\times, ^*/, ^*f)$$

together with a proper elementary embedding $j : \mathbb{R} \to {}^*\mathbb{R}$. This means that j is an elementary embedding and that the image of j is not all of ${}^*\mathbb{R}$. That j is elementary means that for any formula $A(x_1, \ldots, x_n)$ and any reals $r_1, \ldots, r_n$,

$$\mathcal{R} \models A[x_1/r_1, \ldots, x_n/r_n]$$

if and only if

$$^*\mathcal{R} \models A[x_1/j(r_1), \ldots, x_n/j(r_n)].$$

In particular, j is 1-to-1 and $\mathcal{R}$ and ${}^*\mathcal{R}$ satisfy the same sentences.

Notice that $j(0) = {}^*0$ and $j(1) = {}^*1$; in general, we will write $j(r) = {}^*r$. Notice that ${}^*(-r) = {}^* - {}^*r$ for $r \in \mathbb{R}$ and ${}^*(f(r)) = {}^*f({}^*r)$ for $r \in A$.

Let ${}^*\mathcal{R}$ be a nonstandard model. As before, that ${}^*\mathcal{R}$ exists is an easy consequence of the compactness theorem. Say that $r \in {}^*\mathbb{R}$ is *finite* iff there is $n \in \mathbb{N}$ such that ${}^* - n^* < r^* < {}^*n$. Say that r is *infinite* otherwise. Say that r is *infinitesimal* iff for all $n \in \mathbb{N}$ such that $n > 0$, ${}^* - 1/n^* < r^* < {}^*1/n$. Write $r \approx s$ iff $r - s$ is infinitesimal. If $r \in {}^*\mathbb{R}, s \in \mathbb{R}$ and $r \approx {}^*s$, say that s is the *standard part* of r and write $^\circ r = s$.

The following claims are left as exercises:

(i) If $r \in {}^*\mathbb{R}$ is finite then its standard part exists, i.e., there is a unique real number s such that $r \approx s$.

(Recall that a set of reals bounded above has a least upper bound.)

(ii) There are infinite elements in ${}^*\mathbb{R}$. There are infinitesimals (other than ^{*0}). There are infinite natural numbers, i.e., infinite numbers N such that ${}^*\mathbb{N}(N)$.

(iii) The function f is continuous iff for all $x \approx y$, $x, y \in \mathrm{dom}({}^*f)$, ${}^*f(x) \approx {}^*f(y)$. Suppose that $\mathrm{dom}(f) = [0,1]$ and that f is continuous. It follows that if ${}^*0 \mathrel{{}^*<} r \mathrel{{}^*<} {}^*1$ then ${}^*f(r)$ is finite.

(iv) Let N be an infinite natural number. Then

$$\{\, i/N \colon {}^*\mathbb{N}(i) \text{ and } i \mathrel{{}^*\leq} N \,\}$$

is dense in $j([0,1]) = \{\, {}^*s \colon s \in [0,1] \,\}$ and for any $s \in [0,1]$ there is an i/N such that $^\circ(i/N) = s$. Here, ${}^*\leq$ means ${}^*<$ or $=$, and $/$ is really ${}^*/$, but we suppress the $*$ to improve readability.

(v) Suppose from now on that $A = [0,1]$, that f is continuous, and that N is an infinite natural number. The variable i will always stand for a number in ${}^*\mathbb{N}$ such that $i \mathrel{{}^*\leq} N$. For any such i, $^\circ({}^*f(i/N)) = f(^\circ(i/N))$.

(vi) There is an i_0 such that ${}^*f(i_0/N)$ is maximum among

$$\{\, {}^*f(i/N) \colon {}^*\mathbb{N}(i) \text{ and } i^* \leq N \,\}.$$

It follows that $^\circ({}^*f(i_0/N))$ is the maximum of f on $[0,1]$. This shows that a continuous function on a closed interval attains its maximum (and, in particular, is bounded).

(vii) Suppose that $f(0) < 0$ and $f(1) > 0$. Then there is an i such that ${}^*f(i/N)^* \leq {}^*0^* \leq {}^*f((i^* + {}^*1)/N)$. It follows that $^\circ(i/N) = {}^\circ((i^* + {}^*1)/N)$ and therefore $f(^\circ(i/N)) = 0$. This shows the intermediate value theorem.

Chapter 3

Computability and Complexity

3.1 Introduction

Consider a domain D of concrete objects, e.g., numbers, words in an alphabet, finite graphs, etc. We will refer to elements I of D as *instances*.

Definition 3.1.1 Given a set $P \subseteq D$ of instances, the *decision problem* for P is the following question:

> Is there an algorithm (*effective* or *mechanical procedure*) which will tell us, in finitely many steps, for each instance $I \in D$ whether $I \in P$ or $I \notin P$?

Example 3.1.2 Given a formula in propositional logic, is there an algorithm to determine whether it is satisfiable? One can also consider the decision problem where the formulas are now in first-order logic.

Example 3.1.3 Is there an algorithm that, given $n \in \mathbb{N}$ and

$$m \in \{0, 1, \ldots, 9\},$$

tells us whether the n-th digit in the decimal expansion of π is m?

Example 3.1.4 (The Hilbert 10th Problem) Is there an algorithm to verify whether an arbitrary polynomial (in several variables)

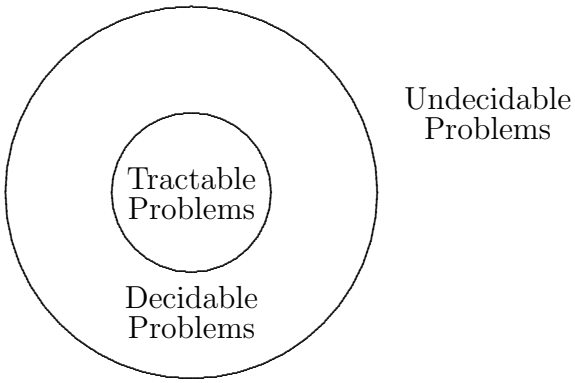
$$\sum a_{i_1 i_2 i_3 \ldots i_n} x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$$

with integer coefficients $a_{i_1 i_2 \ldots i_n}$ has an integer solution?

Definition 3.1.5 If such an algorithm exists for the decision problem (given by) P, we will call P *decidable*. Otherwise we call it *undecidable*.

Example 3.1.6 The validity problem for formulas in propositional logic is decidable (use truth tables). The Hilbert 10th Problem is undecidable (Matyasevich, 1970).

Thus, modulo our rather vague definition of "algorithm," we have the first main classification of (decision) problems as decidable or undecidable. Our next goal will be to classify further the decidable problems according to the (time) complexity of their algorithms and distinguish between those for which an efficient (feasible) algorithm is possible, *the tractable problems*, and those for which it is not, the *intractable* problems.



In particular, this will lead to the study of an important class of problems widely believed to be intractable, but for which no proof has yet been found (this is the famous $\mathcal{P} = \mathcal{NP}$ problem).

We will now formalize these concepts and present an introduction to the study of decision problems and their complexity.

3.2 Decision Problems

We can represent concrete finite objects as words (strings) in a finite alphabet.

Definition 3.2.1 An *alphabet* is an arbitrary nonempty set, whose elements we call *symbols*.

We will be mainly dealing with finite alphabets $A = \{\sigma_1, \ldots, \sigma_k\}$.

Definition 3.2.2 A *word* (or *string*) for A is a finite sequence $a_1 a_2 \ldots a_n$, where each a_i is a symbol in A. If $w = a_1 a_2 \ldots a_n$ is a word, then $n = |w|$ is the *length* of w.

We denote by A^n the set of all words of length n from A. By convention we also have the *empty* word θ which has length 0. Thus $A^0 = \{\theta\}$. Finally we let

$$A^* = \bigcup_n A^n$$

be the set of all words from A (including θ).

Examples 3.2.3

(i) Natural numbers can be represented as words using decimal notation and the alphabet $\{0, 1, 2, \ldots, 9\}$, or in the alphabet $\{0, 1\}$ using binary notation, or even in the alphabet $\{1\}$ using unary ("tally") notation.

(ii) A finite graph $G = \langle V, E\rangle$, with set of vertices $V = \{v_1, \ldots, v_n\}$, can be represented by its adjacency matrix $A = (a_{ij})$, $1 \leq i, j \leq n$, where $a_{ij} \in \{0, 1\}$ and $a_{ij} = 1$ iff $(a_i, a_j) \in E$. This can be in turn represented by the word

$$a_{11} a_{12} \ldots a_{1n} a_{21} \ldots a_{2n} \ldots a_{n1} \ldots a_{nn}$$

in the alphabet $\{0, 1\}$.

Definition 3.2.4 A *decision problem* consists of a finite alphabet A and a set of words $P \subseteq A^*$.

Consider a finite first order language $L = \{f_1, \ldots, f_n;\ R_1, \ldots, R_m\}$. Representing the variable x_n by the string xn, where n is written in binary, we can view every wff in L as a word in the finite alphabet $A = L \cup \{x, 0, 1, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \exists, \forall, (,), =\}$. We let VALIDITY_L be the decision problem corresponding to (A, P), where

$$P = \{S : S \text{ is a sentence in } L \text{ and } \models S\}.$$

3.3 Turing Machines

Now that we have precisely defined decision problems, we will go on to make precise our previously vague concept of an "algorithm." We will do this by presenting a model of computation, using a formalized representation of a computer known as a Turing machine. There are many equivalent formalizations of the notion of "computability," but Turing machines were one of the earliest to be formulated and remain one of the easiest to understand and work with.

As a motivation for the definition which is to follow, consider a running computer program. At any given time, it is in some internal state (for example, which line of the program is being executed) and has some amount of stored data (in variables in memory and on disk). With each "tick" of the computer's clock, it moves to a new state based on its previous state and the contents of its stored data (for example, performing a conditional jump), and may modify the stored data as well.

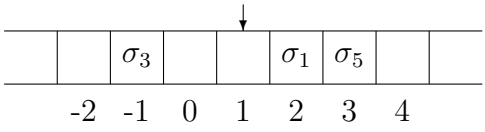
With this characterization in mind, we make the following definition.

Definition 3.3.1 A *Turing machine* M on an alphabet $A = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ consists of the following:

(i) A finite set of states $\{Q_0, Q_1, \ldots, Q_m\}$. We call Q_0 the *start* state and Q_m the *stop* state.

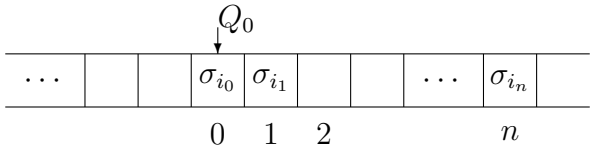
(ii) A table of $(m+1)\cdot(k+1)$ 5-tuples of the form (Q_i, a, b, s, Q_j), where $a, b \in A \cup \{*\}$. (Here $*$ is an additional symbol. whose meaning we explain later. $*$ is sometimes also denoted, for consistency, by $\sigma_0 = *$, so that $A \cup \{*\} = \{\sigma_0, \sigma_1, \ldots, \sigma_k\}$.) For each $0 \le i \le m$, $a \in A \cup \{*\}$, there must be exactly one entry in this table of the form (Q_i, a, b, s, Q_j). Finally we must have $s \in \{-1, 0, +1\}$.

The meaning of the set of states is clear from our discussion above, but the rest of the definition may be somewhat more obscure. Here is how to imagine the operation of a Turing machine. We have a two-way unlimited "tape" divided into squares, each of which may contain at most one symbol in A (or possibly be empty), and a read-write head that at any instant is positioned at exactly one square on the tape.



The head can move in one step along the tape at most one square at a time, right or left, it can write a symbol in the scanned square (replacing any symbol that may have been there) or erase the symbol in the scanned square.

Using this image, for any input $w \in A^*$, we can describe the computation of a Turing machine (TM) on this input w as follows: Let $w = \sigma_{i_0}\sigma_{i_1}\ldots\sigma_{i_n}$. Put this input on the tape as in the picture below and put the scanning head at the leftmost symbol in w (anywhere, if $w = \theta$) and assign to it state Q_0 (the start state):



We now follow the table of the TM by interpreting the entry

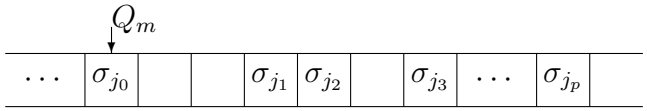
$$(Q_i, a, b, s, Q_j)$$

as expressing the instruction:

When the head is at state Q_i and scans the symbol a (if $a = *$ this means that it scans an empty square – thus $*$ is simply a symbol for the empty square), then prints the symbol b ($b = *$ means "erase what is present at the scanned square"). Then if $s = +1$, move the head one square to the right; if $s = -1$, move one square to the left; and if $s = 0$, leave the head where it is. Finally, change state to Q_j.

So in terms of our original characterization of a computer program, the symbol a represents the possible role of stored data in choosing the new state, while b is the possible change to the stored data. s is an artifact of how we have chosen to represent the stored data as an infinite tape. While it may seem like a limitation to be only able to examine and change one square of the tape at a time, this can in fact be overcome by the addition of a suitable number of intermediate internal states and transitions, and it simplifies the analysis considerably.

The computation proceeds this way step by step, starting from the input w. It terminates exactly when it reaches the stop state, Q_m. When the computation halts, whatever string of symbols from A remain on the tape to the right of the scanning head, say $v = \sigma_{j_0}\sigma_{j_1}\ldots\sigma_{j_p}$ (from left to right), we call the string v the *output* of the computation.



Note that a computation from a given input w may not terminate, because we need not ever reach the stop state Q_m.

Example 3.3.2 Consider the following TM on the alphabet $A = \{0, 1\}$:

states: Q_0, Q_1, Q_2

table:
$(Q_0, a, \bar{a}, +1, Q_0)$, for $a \in A$, where $\bar{a} = 1 - a$,
$(Q_0, *, *, -1, Q_1)$
$(Q_1, b, b, -1, Q_1)$, for $b \in A$,
$(Q_1, *, *, +1, Q_2)$
$(Q_2, a, a, 0, Q_2)$, $a \in A \cup \{*\}$

On an input of, say, $w = 1011101$, the machine terminates with output 0100010. In general, on input $w \in A^*$ the machine terminates with output $\bar{w}$ which is equal to w with 0s and 1s switched. (This operation is also known as the binary "ones complement".)

Example 3.3.3 $A = \{1\}$

states: Q_0, Q_1

table:
$(Q_0, 1, 1, +1, Q_0)$
$(Q_0, *, 1, +1, Q_0)$
$(Q_1, a, a, 0, Q_1)$, $a \in A \cup \{*\}$

Then no matter what the starting input is, say $w = 1\ldots1$ in A^*, this machine does not terminate: it just keeps adding 1's to the right of w forever.

Definition 3.3.4 We call a decision problem (A, P) *decidable* if there is a Turing machine M on some finite alphabet $B \supseteq A \cup \{Y, N\}$ such that for every word $w \in A^*$:

$w \in P \Rightarrow$ on input w the machine M halts with output Y
$w \notin P \Rightarrow$ on input w the machine M halts with output N

Otherwise, P is called *undecidable*.

3.4 The Church-Turing Thesis

We have now formalized the concept of decidability using the Turing machine model of computation. In this section we will discuss alternative ways to formalize this concept, and see that they are all equivalent.

3.4.A Register Machines

First we will discuss in detail one more model of computation which is closer to the operation of real-world computers: the so-called *register machines*.

Definition 3.4.1 A *register machine* N on an alphabet $A = \{\sigma_1, \ldots, \sigma_k\}$ is a program consisting of a finite list $1 : I_1, \ldots, n : I_n$ of consecutively numbered instructions, each of which is one of the following types:

ADD$_j$ Rm,	$1 \leq j \leq k$; $m = 1, 2, \ldots$
DEL Rm,	$m = 1, 2, \ldots$
CLR Rm,	$m = 1, 2, \ldots$
R$p \leftarrow$Rm,	$m, p = 1, 2, \ldots$
GO TO ℓ,	$1 \leq \ell \leq n$
IF FIRST$_j$ Rm GO TO ℓ	$1 \leq j \leq k$; $1 \leq \ell \leq n$; $m = 1, 2, \ldots$
STOP	

We assume moreover that $I_i = \text{STOP}$ if and only if $i = n$.

To understand the operation of this machine, imagine an infinite list of registers $R1, R2, \ldots$, each of which is capable of storing an arbitrary word in A^*. Then the meaning of each instruction in the above list is the following:

ADD$_j$ Rm:	add σ_j to the right of (the word in) Rm
DEL Rm:	delete the leftmost symbol in Rm
CLR Rm:	erase the word in Rm
R$p \leftarrow$Rm:	replace the word in Rp by the word in Rm
GO TO ℓ:	go the ℓth instruction $\ell : I_\ell$
IF FIRST$_j$ Rm GO TO ℓ:	if the word in Rm starts with σ_j, go to the ℓth instruction; otherwise go to the next instruction

STOP: halt the computation.

Using this, we can now describe the computation of the register machine (RM) N on any input $w \in A^*$: Put this input w in the first register R1 (with all other registers empty). Follow the instructions $1 : I_1, \ldots, \ell : I_\ell$ as above in order (unless of course a GO TO instruction is met). This computation terminates exactly when we reach (if ever) I_n : STOP. If $v \in A^*$ is the word in the register R1, when the computation stops, we call v the *output* of the computation. Again a computation from a given input w may not terminate. Also note that if the program for N mentions at most the registers R1, R2,..., Rm, then these are the only registers used in the computation on any input.

Example 3.4.2 Consider the following RM on the alphabet $A = \{1, \diamond\} = \{\sigma_1, \sigma_2\}$

1: IF FIRST$_1$ R1 GO TO 5
2: DEL R1
3: IF FIRST$_1$ R1 GO TO 8
4: GO TO 11
5: DEL R1
6: ADD$_1$ R2
7: GO TO 1
8: DEL R1
9: ADD$_1$ R2
10: GO TO 3
11: R1←R2
12: STOP

Then on input $\underbrace{1\ 1\ldots1}_{n} \diamond \underbrace{1\ 1\ldots1}_{m}$ the machine terminates with output $\underbrace{1\ 1\ldots1}_{n+m}$ $(n, m \geq 1)$. Thus it performs addition in unary notation.

Example 3.4.3 Consider the alphabet $A = \{1\}$, and the machine:

1: ADD$_1$ R1
2: GO TO 1
3: STOP

Then this RM does not terminate on any input in A^*.

Although the TM and RM models of computation are quite different, it can be shown that they are equivalent, in the sense that whatever can be computed using one of the models can be computed with the other, and vice versa. More precisely, consider two finite alphabets A, B and a *partial function* $f : A^* \to B^*$, that is a function whose domain is a subset of A^* and which takes values in B^*. (Normally, when we write $g : X \to Y$ we mean that g is a *total* function with domain *exactly* X and values in Y. We will however use this notation below even if the domain of g is only a (possibly proper) *subset* of X, but in this case we will explicitly state that g is partial.)

Definition 3.4.4 $f : A^* \to B^*$ is TM-*computable* if there is a finite alphabet $C \supseteq A \cup B$ and a Turing Machine M on C such that for each input $w \in A^* (\subseteq C^*)$, M terminates on w iff w is the domain of f (i.e. $f(w)$ is defined), and in that case the output of M on input w is $f(w)$.

We define what it means for f to be RM-computable in a similar way. We can then show

Theorem 3.4.5 *TM-computable = RM-computable.*

This is a programming exercise and we will not discuss it in detail here. First, we show that given a RM N on an alphabet C, one can construct a TM M on the alphabet $D = C \cup \{\diamond\}$, such that if we put $w \in C^*$ as input in N and the same input $w \in C^*$ in M, then N terminates on w iff M terminates on w, and for such w the outputs of N and M are the same.

Specifically, we assume that all the registers mentioned in N are among R1, R2,..., Rm. If R1,..., Rm, at some stage in the computation, contain the words $w_1, \ldots, w_m$, respectively, then the idea is to represent their contents by the word $w_1 \diamond w_2 \diamond \cdots \diamond w_m \diamond$ in the tape of the TM M, with the head scanning the leftmost symbol of w_1 (or the first $\diamond$ if $w_1 = \theta$). Then for each instruction of N we will introduce a block of entries in the table of M, whose effect on $w_1 \diamond w_2 \diamond \cdots \diamond w_m \diamond$ will correspond to the effect of the instruction of N on the corresponding register values. In the case of GOTO instructions this simulation will preserve the same flow of control.

Conversely, we can show that given a TM M on an alphabet C, one can construct a RM N on some alphabet $D \supseteq C$ such that again if we put $w \in C^*$ as input in M and N, then M terminates on w iff N terminates on w, and for such w the outputs of M and N on w are the same. For example, one way to do that is to take $D = \{\sigma_0, \sigma_1, \ldots, \sigma_k\} \cup \{Q_0, \ldots, Q_m\}$,