Computability II

Readings: Sipser, Chapter 4.

Theorem 2 Every language in NP is decidable.

PROOF: Let $L \in$ NP, and let $R(x, y)$ be a relation decidable by a TM M in time $|x|^c$, for some constant c, such that:

$$x \in L \Longleftrightarrow \exists y R(x, y).$$

Idea: try each y of length at most $|x|^c$ and see if $R(x, y)$ is true. More precisely, a TM M' deciding L does the following:

```
On input x,
for each y ∈ Σ* such that |y| ≤ |x|^c do
    evaluate R(x,y) by simulating M on (x,y),
    if R(x,y) is true, then
       Accept
    end if
end for
Reject /* no certificate y was found */
```

This finishes the proof. □

Relationship among the classes of languages that we have seen so far:

$$\text{P} \subseteq \text{NP} \subset \text{decidable} \subset \text{semi-decidable}$$

We know that P $\subseteq$ NP, but we do not know if P $\neq$ NP. We do know however that NP $\neq$ decidable, and decidable $\neq$ semi-decidable.

Here is a language that is decidable, but is provably not in NP.

Presburger Arithmetic: First-order theory of $[\mathbb{N}, +, \leq, 0, 1]$.

Question: Given a formula built from $\forall, \exists, \wedge, \vee, \neg, x, y, z, \ldots$ (ranging over $\mathbb{N}$), $+, \leq, 0, 1$, is this formula true in $\mathbb{N}$?

For example, the following formula in the language of Presburger Arithmetic expresses the fact that every natural number is either even or odd (so this formula is true in $\mathbb{N}$):

$$\forall x \exists y [x = y + y \vee x = (y + y) + 1]$$

Let L_{PA} be the language of all formulas of Presburger Arithmetic that are true in $\mathbb{N}$. Then the above question can be restated as follows: Is L_{PA} decidable?

It can be proved that L_{PA} is decidable (in time $2^{2^{2^{cn}}}$). On the other hand, it can be proved that any TM deciding L_{PA} must spend time at least $2^{2^{c'n}}$. This shows that L_{PA} is not in NP, since every language in NP is decidable in single-exponential time, 2^{n^d}.

The Halting Problem

We can encode any string over an arbitrary alphabet Σ_1 by using the fixed alphabet $\Sigma = \{0, 1\}$.

For example, let $\Sigma_1 = \{a, b, c\}$. Encode a as 01, b as 011, and c as 0111. Then $aabc$ becomes 01010110111. Note that decoding is unique.

We can also encode the different states of a TM and the transition function δ using the alphabet Σ.

Thus, we can encode arbitrary TM's M and their inputs w by using the alphabet $\Sigma = \{0, 1\}$. Let $\langle M, w\rangle$ denote such an encoding of a TM M and its input w.

Let

$$A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\ \}.$$

Theorem 3 A_{TM} is not decidable.

A_{TM} is however semi-decidable by the *universal* TM U as follows:

> `On input` $\langle M, w\rangle$`, where` M `is a TM and` w `is a string,` U `simulates` M `on input` w`, accepting whenever` M `accepts, and rejecting whenever` M `rejects.`

The TM U behaves as an *interpreter*. The existence of the universal TM was proved by Turing (1936); the proof is not hard, but requires some care.

Diagonalization

The technique used to show the undecidability of A_{TM} was first used by Georg Cantor (1873) to show $\mathbb{R}$ is not countable.

Definition 3 A set A is *countable* if A is finite, or if there is a one-one and onto function $f : \mathbb{N} \to A$.

If A is an infinite countable set, then for some function f, $A = \{f(1), f(2), \dots\}$.

Example: The set $\mathbb{Q}$ of rationals is countable. Consider an infinite matrix (a_{ij}), where $a_{ij} = \frac{i}{j}$. We can list all the rationals by traversing the diagonals of the matrix, omitting the elements that would cause a repetition.

Theorem 4 (Cantor) The set $\mathbb{R}$ of reals is not countable.

Proof (by contradiction): Suppose $f : \mathbb{N} \to \mathbb{R}$ is one-one and onto. This means that all the reals between 0 and 1 will appear in the following table:

n	the fractional part of $f(n)$
1	$0.\underline{a_1}a_2a_3a_4\ldots$
2	$0.b_1\underline{b_2}b_3b_4\ldots$
3	$0.c_1c_2\underline{c_3}c_4\ldots$
$\vdots$	$\ddots$

We will construct a real number x between 0 and 1 such that $x \neq f(1)$, $x \neq f(2)$, $x \neq f(3)$, and so on. Define $x = 0.x_1x_2x_3x_4...$ so that $x_1 \neq a_1$, $x_2 \neq b_2$, $x_3 \neq c_3$, and so on. Namely, if $a_1 = 1$, then set $x_1 = 2$; if $a_1 \neq 1$, then set $x_1 = 1$. Similarly for b_2 and x_2, c_3 and x_3, and so on.

By construction, $x \neq f(n)$ for every $n \in \mathbb{N}$ since x differs from $f(n)$ in the nth fractional digit. Hence x is not in $\mathbb{R}$, which is a contradiction.

Proof of Theorem 3: In this proof we are stressing the analogy to Cantor's diagonal argument. Our proof is by contradiction. Suppose that a TM H decides A_{TM}. That is,

$$H(\langle M, w\rangle) = \begin{cases} \text{Accept} & \text{if } M \text{ accepts } w \\ \text{Reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

Let $M_1, M_2, M_3, \ldots$ be a list of all TM's. Consider an infinite matrix (h_{ij}), where $h_{ij} = H(\langle M_i, \langle M_j\rangle\rangle)$. For example, the top left corner of such a matrix could look as follows:

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\cdots$
M_1	$\underline{\text{accept}}$	reject	reject	
M_2	reject	$\underline{\text{reject}}$	accept	$\cdots$
M_3	accept	accept	$\underline{\text{accept}}$	
$\vdots$		$\vdots$		$\ddots$

Assuming the existence of H, we can define a TM D that differs from each M_i. We define D to behave as follows:

> On input $\langle M\rangle$, where M is a TM, D simulates H on $\langle M, \langle M\rangle\rangle$ and outputs the opposite of what H outputs.

That is,

$$D(\langle M\rangle) = \begin{cases} \text{Accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \text{Reject} & \text{if } M \text{ accepts } \langle M\rangle \end{cases}$$

Note that D is a TM since H is. But now consider what D does on its own encoding:

$$D(\langle D\rangle) = \begin{cases} \text{Accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \text{Reject} & \text{if } D \text{ accepts } \langle D\rangle, \end{cases}$$

which is a contradiction. Hence H cannot exist, and so A_{TM} is undecidable.

Theorem 3 applies to *every* general programming language. That is, the Halting Problem for every general programming language is undecidable.

Finally, consider the following question: Can there exist a TM that makes finitely many mistakes, but otherwise decides A_{TM}?

The answer is no. We can always "patch up" a TM making finitely many mistakes by "hard-coding" the correct answers to the problematic inputs into the finite control of the TM. But then we would obtain a TM that makes no mistakes in deciding A_{TM}, which is impossible by the theorem we just proved.

Reducibility

Recall that if $f : \Sigma^* \to \Sigma^*$, M is a TM, we defined the notion "M computes f": M on input x computes $f(x)$ and halts. M must halt on every input.

Definition 4 f is computable if some TM computes f.

Definition 5 If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$, we say that L_1 is *many-one reducible* to L_2 (denoted $L_1 \leq_m L_2$) iff there is a computable function $f : \Sigma^* \to \Sigma^*$, s.t. $x \in L_1 \Leftrightarrow f(x) \in L_2$, for all $x \in \Sigma^*$.

Note that $L_1 \leq_p L_2 \Rightarrow L_1 \leq_m L_2$. Also recall that: $L_1 \leq_p L_2$ and $L_2 \in P \Rightarrow L_1 \in P$. We have an analogous result for decidable sets:

Theorem 5 If $L_1 \leq_m L_2$ and L_2 is decidable, then L_1 is decidable.

PROOF: Suppose M decides L_2. Then M' decides L_1, where M' is given by: on input x, compute $f(x)$, then run M on input $f(x)$. $\square$

Theorem 6 If $L_1 \leq_m L_2$ and L_2 is semi-decidable, then L_1 is semi-decidable.

PROOF: Same as above. $\square$

Corollary 1 If $L_1 \leq_m L_2$, and L_1 is not decidable, then L_2 is not decidable.

PROOF: This is the contrapositive of the above theorem (the contrapositive of $A \Rightarrow B$ is $\neg B \Rightarrow \neg A$, and they are logically equivalent). $\square$

Thus, to show that L is not decidable, it suffices to show $A_{\mathrm{TM}} \leq_m L$.

Analogy: If $L_1 \leq_p L_2$ and L_1 is NP-hard, then L_2 is NP-hard.

Define $\mathrm{HALT}_{\mathrm{TM}} = \{\langle M, w\rangle | M \text{ halts on input } w\}$.

Lemma 1 $\mathrm{HALT}_{\mathrm{TM}}$ is not decidable.

PROOF: We show that $A_{\mathrm{TM}} \leq_m \mathrm{HALT}_{\mathrm{TM}}$. Given an input $\langle M, w\rangle$ to A_{TM} we construct an input $f(\langle M, w\rangle) = \langle M', w'\rangle$ to $\mathrm{HALT}_{\mathrm{TM}}$, s.t. $\langle M, w\rangle \in A_{\mathrm{TM}} \Leftrightarrow \langle M', w'\rangle \in \mathrm{HALT}_{\mathrm{TM}}$, i.e. M accepts $w \Leftrightarrow M'$ halts on w'. Take $w' = w$. Let M' be a modified version of M, namely, M' is the same as M, except M' loops when M hits a reject state. To do so, replace q_{reject} by a new state q', which loops. Therefore, $\mathrm{HALT}_{\mathrm{TM}}$ is undecidable (but semi-decidable). $\square$