

# Chapter 5

## Decidability

We have introduced Turing Machine as a model for the general computation and defined the informal notion of algorithms in terms of Turing Machine. Now, we want to investigate the power of such a model to solve problems. Particularly, we will demonstrate problems that can be solved under this model and those that can't. For example, we will show that there exists an algorithm that will decide if a CFL will generate a string, which is the centerpiece of a compiler.

We will also show that there doesn't exist any algorithm that will tell us if an arbitrary TM will accept an arbitrary string. To prove that a problem is algorithmically unsolvable will enforce us to its simplification. It will also lead us to a better understanding of its nature.

## Decidable problems w.r.t. RL

The *accepting problem for DFAs* of testing if a particular DFA accepts a given string can be expressed as the following language:  $A_{\text{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts } w. \}$ . The notions of  $A_{\text{NFA}}$  and  $A_{\text{REX}}$  are similarly defined.

**Theorem:**  $A_{\text{DFA}}$  is a decidable language.

**Proof:** For any input  $\langle B, w \rangle$ , where  $B$  is a DFA and  $w$  is a string:

1. Simulate  $B$  on  $w$ .
2. If the simulation ends in an accept state, accept it; otherwise, rejects it.  $\square$

Because we can convert any NFA, or a regular expression, to a DFA, thus, we have the following result:

**Theorem:** Both  $A_{\text{NFA}}$ , and  $A_{\text{REX}}$  are decidable languages.

Let  $E_{\text{DFA}}$  be  $\{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \}$ .

**Theorem:**  $E_{\text{DFA}}$  is decidable.

**Proof:** A DFA accepts a string iff it is possible to travel from the start state to an accept state. Thus, we can construct the following TM:

For any input  $\langle A \rangle$ , where  $A$  is a DFA,

1. Mark the start state of  $A$ .
2. Repeat until no new state gets marked
3. Mark any state that has a transition coming into it from a marked state.
4. If no accept state gets marked, accept; otherwise, reject.  $\square$

Let  $EQ_{\text{DFA}}$  be  $\{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$ .

**Theorem:**  $EQ_{\text{DFA}}$  is decidable.

**Proof:** We will construct another DFA  $C$ , such that  $C$  only accepts those strings that are accepted by either  $A$  or  $B$ , but not by both. Then,  $L(C) = \emptyset \equiv L(A) = L(B)$ . Obviously,

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (L(B) \cap \overline{L(A)}).$$

Because both  $L(A)$  and  $L(B)$  are RL's, and RLs are closed under union, intersection, as well as complementation, so is  $L(C)$ .

Thus, to decide if  $L(A) = L(B)$ , we simply construct  $C$  and run  $E_{\text{DFA}}$  on  $C$ .  $\square$

**Homework:** Exercise 4.1-4.3 in pp. 150–151.

## Decidable problems w.r.t. CFL

Let  $A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that accepts } w. \}$ .

**Theorem:**  $A_{\text{CFG}}$  is a decidable language.

**Proof:** The idea to try all derivations does not work, in general, since if  $G$  doesn't generate  $w$ , the algorithm might never halt. On the other hand, we can always convert  $G$  to a Chomsky Normal Form. If the latter generates  $w$ , it must generate it in  $2|w| - 1$  steps. Thus, we have the following "inefficient" compiler.

For any input  $\langle G, w \rangle$ , where  $G$  is a CFG and  $w$  is a string:

1. Convert  $G$  to an equivalent grammar in Chomsky normal form.
- 2 List all derivations with  $2|w| - 1$  steps.
3. If any of them accepts  $w$ , accept it; otherwise, reject. □

Let  $E_{CFG}$  be  $\{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$ .

**Theorem:**  $E_{CFG}$  is decidable.

**Proof:** We can't use the previous result to test for emptiness, as there are infinite number of strings over any alphabet.

The following is the algorithm:

For any input  $\langle G \rangle$ , a CFG,

1. Mark all terminals in  $G$ .
2. Repeat until no new variable gets marked.
3. Mark any variable  $A$ , where  $G$  contains a rule  $A \rightarrow U_1 \cdots U_k$ , and all  $U_i$ 's have been marked.
4. If the start symbol is not marked, accept; otherwise, reject.  $\square$

Let  $EQ_{CFG}$  be  $\{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$ .

We can't use the same idea as we used to show that  $EQ_{DFA}$  is decidable. (Why?) In fact,  $EQ_{CFG}$  is not decidable.

**Theorem:** Every CFL is decidable.

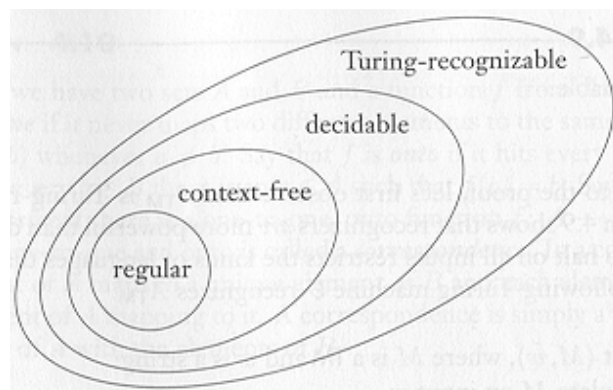
This result means that for any CFL,  $L$ , and any string  $w$ , there is a TM that decides if  $w \in L$ .

**Proof:** Let  $G$  be a CFG for  $L$ , and let  $S$  be the TM we used to decide the acceptance problem, we construct a TM,  $M_G$  : For any input  $\langle G, w \rangle$ ,

1. Run TM  $S$  on  $\langle G, w \rangle$ .
2. If  $G$  accepts  $w$ , accept; otherwise, reject.  $\square$

## Some positive results

So far, we have coined four classes of languages: regular, context-free, enumerable and decidable. The following chart shows their relationship.



**Homework:** Exercises 4.4, and 4.11.

# The Halting problem

We prove the existence of a concrete, and easily understood, problem that is algorithmically unsolvable. By the halting problem, we mean the following: *given a Turing Machine,  $M$ , and an input string,  $s$ , whether  $M$  accepts  $s$ ?* Let's call it  $A_{\text{TM}}$ .

**Theorem:**  $A_{\text{TM}}$  is undecidable.

Obviously, it is enumerable, i.e., there is TM that can *accept*  $A_{\text{TM}}$ . For example, we can construct the following TM,  $U$ , the *universal Turing Machine*, for input  $\langle M, w \rangle$ , where  $M$  is a TM and  $w$  is an input string

1. Simulate  $M$  on input  $w$ ,
2. If  $M$  ever enters its accept state, accept; if  $M$  ever enters its reject state, reject.  $\square$

# The diagonalization

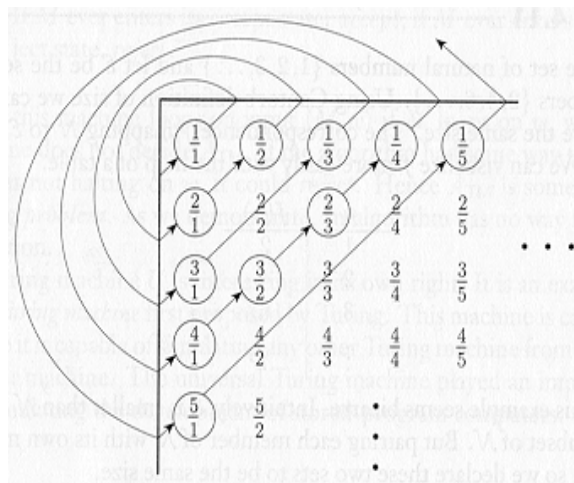
This technique was originally used to investigate infinite sets. Given two finite sets, it is easy to see which is larger. But, it is not so easy to answer the same question for two sets that contain infinite amount of elements. Cantor observed that if two finite sets are of the same size, then there exists a 1-1 correspondence between the elements of these two sets and proposed to use the same criterion for infinite sets.

**Definition:** Assume that we have two sets  $A$  and  $B$  and a function  $f : A \mapsto B$ . By saying that  $f$  is *one to one*, we mean that  $f$  never maps to different elements to the same element, i.e.,  $a \neq b \Rightarrow f(a) \neq f(b)$ ;  $f$  is *onto*, if  $f$  hits every element in  $B$ , i.e.,  $\forall b \in B, \exists a, f(a) = b$ ;  $A$  and  $B$  are of the *same size*, if there is a one to one, and onto function  $f : A \mapsto B$ . (?)

## Some examples

Let  $\mathcal{N}$  be the set of natural numbers,  $\{1, 2, \dots\}$  and let  $\mathcal{E}$  be the set of all the even natural numbers,  $\{2, 4, \dots\}$ . Which contains more numbers? Intuitively,  $\mathcal{N}$  does. However, because of the existence of  $f : \mathcal{N} \mapsto \mathcal{E}$ ,  $\forall n \in \mathcal{N}, f(n) = 2n$ , and the fact that  $f$  is both one-to-one and onto, the two sets are of the same size.

Let  $\mathcal{Q}$  be the set of all the rational numbers,  $\{\frac{m}{n} | m, n \in \mathcal{N}\}$ . Even though  $\mathcal{Q}$  appears to be much larger than  $\mathcal{N}$ , once again, they are actually of the same size, because of the following correspondence.



**Definition:** A set is *countable* if either it is finite or it has the same size as  $\mathcal{N}$ .

**Question:** Is it true all the infinite sets are countable?

**A short answer:** No! In fact, if the answer were positive, cantor's theory would not be interesting, at all.

**A longer one:** Let  $\mathcal{C}$  be the set of all the real numbers, i.e., the numbers that have decimal representation, e.g.,  $3.1415926\dots$  etc.  $|\mathcal{C}| > |\mathcal{N}|$ .

**Theorem:**  $\mathcal{C}$  is not countable.

**Proof by contradiction:** Just assume that  $\mathcal{C}$  is countable. By definition, there exists a one-to-one, onto function,  $f$ , such that it will map every natural number to a unique real number, and, for every real number, there is a corresponding natural number.

We construct a real number as follows:  $x = 0.x_1x_2\cdots x_i\cdots$  such that for all  $n \in \mathcal{N}$ ,  $x_n \neq$  the  $n^{\text{th}}$  digit of  $f(n)$ . As  $x$  is obviously a real number, by definition, there must be a natural number  $n_x$ , such that  $f(n_x) = x$ . However, by construction, the  $n_x^{\text{th}}$  digit of  $x$  is not the same as the  $n_x^{\text{th}}$  digit of  $f(n_x) = x$ . This is a contradiction. Thus, there does not exist such a function  $f$ , i.e.,  $\mathcal{C}$  is not countable.  $\square$

**Homework:** 4.6-4.9 in pp. 151.

The key idea in the above proof uses the diagonal line of the list, thus the name. This technique is one of the few important ones that are frequently used in the study of the theory of computation.

The above result can be used to show that there exists languages that are not enumerable. We will show first that the set of Turing Machines is countable, but the set of all the languages are not. As each TM accepts only one language, there must be some language that is not accepted by any TM, thus, by definition, not enumerable.

**Corollary:** There are languages that are not enumerable.

**Proof:** To show that the set of TMs is countable, we merely observe that the set of all strings of  $\Sigma^*$  for any finite alphabet  $\Sigma$  is countable. With only finite number of strings of a given length, we can list all strings of length 0, followed by all strings of length 1, etc..

The set of all the TMs is countable, since any TM can be coded into a string. If we just cross over those strings that don't represent a TM, we have a list of all the TMs.

To show that the set of languages is not countable, we observe that for any  $\Sigma$ , although  $\Sigma^*$  is countable,  $\mathcal{P}(\Sigma^*)$  is not, by establishing a one-to-one and onto correspondence with the set of all the infinite sequences consisting of 0 and 1. (?)

# The main result

Now, we show the diagonalization technique to show that  $A_{\text{TM}}$  is not decidable.

**Proof:** Just assume it is decidable and  $H$  is its decider. Hence,

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w, \\ \text{reject} & \text{otherwise.} \end{cases}$$

Now, we use  $H$  as a subroutine to construct a new TM,  $D$ , which determines what  $M$  does when the input to  $M$  is its own description,  $\langle M \rangle$ .

Once obtaining this information,  $D$  does the opposite: For any input  $\langle M \rangle$ ,

1. Run  $H$  on input  $\langle M, \langle M \rangle \rangle$ .
2. If  $H$  accepts, rejects; otherwise, accepts.

Hence, we have that

$$D(\langle M \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ rejects } \langle M \rangle, \\ \text{reject} & \text{otherwise.} \end{cases}$$

**Question:** What happens when we run  $D$  with its own description,  $\langle D \rangle$ ?

**Answer:** We must have

$$D(\langle D \rangle) = \begin{cases} \text{accept } \langle D \rangle & \text{if } D \text{ rejects } \langle D \rangle, \\ \text{reject } \langle D \rangle & \text{otherwise.} \end{cases}$$

In other words,

$$D \text{ accepts } \langle D \rangle \text{ iff } D \text{ rejects } \langle D \rangle .$$

Thus, there doesn't exist such a  $D$ , which implies that there doesn't exist a decider  $H$  for the original Halting problem, either.  $\square$

## Where is the beef?

Where did we use the diagonalization in the above proof? The following charts assume what  $M_i$  will do with  $\langle M_i \rangle$ , gives the value of  $H(M_i, \langle M_i \rangle)$ ,  $D$ 's behavior and where the contradiction occurs.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\dots$
$M_1$	Accept		Accept		
$M_2$	Accept	Accept	Accept	Accept	
$M_3$					
$M_4$	Accept	Accept			

Entry  $(i, j)$  is *accept* if  $M_i$  accepts  $\langle M_j \rangle$  .

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\dots$
$M_1$	Accept	reject	Accept	reject	
$M_2$	Accept	Accept	Accept	Accept	
$M_3$	reject	reject	reject	reject	
$M_4$	Accept	Accept	reject	reject	

Entry  $(i, j)$  is the value of  $H$  on input  $\langle M_i, \langle M_j \rangle \rangle$  .

## Here it is!

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	Accept	reject	Accept	reject		Accept
$M_2$	Accept	Accept	Accept	Accept		Accept
$M_3$	reject	reject	reject	reject		reject
$M_4$	Accept	Accept	reject	reject		Accept
⋮						
D	reject	reject	Accept	Accept		?
⋮						

By construction,  $D$  accepts  $\langle M_1 \rangle$  since  $M_1$  accepts  $\langle M_1 \rangle$ , but rejects  $\langle M_2 \rangle$  since  $M_2$  rejects  $\langle M_2 \rangle$ , .... The question is what to do with  $\langle D \rangle$ ?

Again by construction,  $D$  should accept  $\langle D \rangle$ , if  $D$  rejects  $\langle D \rangle$ ; and reject  $\langle D \rangle$ , if  $D$  accepts  $\langle D \rangle$ . In terms of the picture, we should fill in "Accept" in the '?' iff we should fill in it with "reject".

Thus, a contradiction occurs at "?".

## Near the end

The complement of a language is the collection of all the strings that don't belong to the original language. We call an enumerable language *co-enumerable* if it is the complement of an enumerable language.

**Theorem:** A language is decidable iff it is both enumerable and co-enumerable.

**Proof:** As any decidable language is also enumerable and the complement of a decidable language is also decidable, the “only-if” part is done.

Let  $M_1$  and  $M_2$  be TMs that accept a language,  $A$ , and its complement, we construct the following TM for the “if” part. For any input  $w$ ,

1. Run both  $M_1$  and  $M_2$  on input  $w$  in the “dovetail” style.
2. If  $M_1$  accepts, accept; if  $M_2$  accepts, reject.

As  $w$  either belongs to  $A$  or  $\overline{A}$ , thus, either  $M_1$  or  $M_2$  will accept  $w$ . Whenever  $M_1$  or  $M_2$  halts,  $M$  halts, when  $M$  accepts everything in  $A$  and rejects anything in  $\overline{A}$ , thus,  $M$  is a decider for  $A$ .  $\square$

We once showed that because the set of TMs is countable, while the set of all languages is not, there must exist a language that is not enumerable. Now, we show a concrete example.

**Corollary:**  $\overline{A_{\text{TM}}}$  is not enumerable.

**Proof:** We know that  $A_{\text{TM}}$  is enumerable. If its complement were enumerable also, it would be decidable.  $\square$