CS340: Theory of Computation Sem I 2017-18

Lecture Notes 17: Undecidability

Raghunath Tewari IIT Kanpur

We shall prove that not all problems/languages are decidable. In other words there exists languages that are not accepted by any halting Turing machine. The proof uses the diagonalization method to construct a language and then argue that there does not exist any TM that accepts this language.

1 A Non Turing Recognizable Language

We use the diagonalization argument to show the existence of a non Turing recognizable language. First let us lay down a few notations and conventions that will be required for our proof.

- Every TM has a finite description (states, transition function, etc.).
- Description of a TM can be thought of as a finite string over $\{0,1\}$. This can be achieved in many ways. If the description contains k different symbols, then for each symbol associate a string over $\{0,1\}$ of length $\lceil \log k \rceil$. Now replace every symbol by its corresponding bit string. Hence we get a string over $\{0,1\}$ for every TM. We denote this description of a TM M as $\langle M \rangle$.
- Using the bit string representation described earlier, we can associate a natural number to every bit string in the following natural way. Append a 1 to the left of the string (to distinguish between preceding 0's) and convert it to decimal.
- If a natural number does not correspond to any TM then associate it with some fixed TM (say a TM that accepts all strings).
- By the above scheme, every natural number corresponds to some TM and for every TM there is at least one natural number representing it. For $i \geq 1$, let M_i denote the i-th TM. This gives us a surjective function from natural numbers to the set of *all* TMs.

Theorem 1. *There exists a language that is not Turing recognizable.*

Proof. Let $s_1, s_2, \ldots$ be the lexicographic ordering of all binary strings. Define

$$L_d = \{s_i \mid M_i \text{ does not accept } s_i\}.$$

Suppose L_d is Turing recognizable. Then there is a TM M_j such that $L_d = L(M_j)$ (that is, M_j is the j-th TM in the sequence described above). Now,

$$M_j \text{ accepts } s_j \iff s_j \in L(M_j) \iff s_j \in L_d \iff M_j \text{ does not accept } s_j.$$

Therefore we have a contradiction and L_d is not Turing recognizable. □

2 Undecidable Languages

Consider the language

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

It is easy to show that A_{TM} is Turing recognizable. Given an input $\langle M, w\rangle$, simulate M on w. If M accepts w then *accept* and if M rejects w then *reject*.

Theorem 2. *A_{TM} is undecidable.*

Proof. Suppose A_{TM} is decidable. Then there exists a halting TM H that on input $\langle M, w\rangle$, *accepts* if M accepts w and *rejects* if M does not accept w.
Using H, construct another machine N as follows:

Description of Turing machine N

Input: $\langle M\rangle$ where M is a TM

1. Simulate H on $\langle M, \langle M\rangle\rangle$. (A description of H is hardcoded into the machine N.)
2. If H rejects then *accept* else *reject*.

Note that the machine H is a halting TM, hence N is also a halting TM. Now consider what happens when the machine N is provided with the input $\langle N\rangle$.

$$N \text{ accepts } \langle N\rangle \iff H \text{ rejects } \langle N, \langle N\rangle\rangle \iff N \text{ does not accept } \langle N\rangle$$

This is a contradiction. Hence A_{TM} is undecidable. □

2.1 Halting Problem

We can show that a language is undecidable by "reducing" a known undecidable language to the given language. Later we will formalize the definition of reduction. But for now let us show the undecidability of the *halting problem* by proving that if the halting problem is decidable then A_{TM} is also decidable. The halting problem is defined as

$$H_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}.$$

Suppose H_{TM} is decidable via a halting TM H. Using H we will construct a halting TM H' for A_{TM} as follows:

Description of Turing machine H'

Input: $\langle M, w\rangle$ where M is a TM and w is a string

1. Simulate H on $\langle M, w\rangle$.
2. If H rejects $\langle M, w\rangle$ then *reject*.
3. If H accepts then simulate M on w.
   - If M accepts then *accept* and if M rejects w then *reject*.

Observe that when we simulate M on w in the above algorithm it will always halt. This is because if M does not halt on w then H rejects in Step 2 itself and we do not proceed to Step 3.

Exercise 1. Show that if A_{TM} is decidable then H_{TM} is decidable (essentially converse of the above construction).

Exercise 2. Show that H_{TM} is Turing recognizable.

Note 1. $\overline{A_{TM}}$ is not Turing recognizable. This is because we know that A_{TM} is Turing recognizable and if $\overline{A_{TM}}$ is also Turing recognizable then it implies that A_{TM} is decidable. The same is true for H_{TM} as well.

This can be generalized to give the following Proposition.

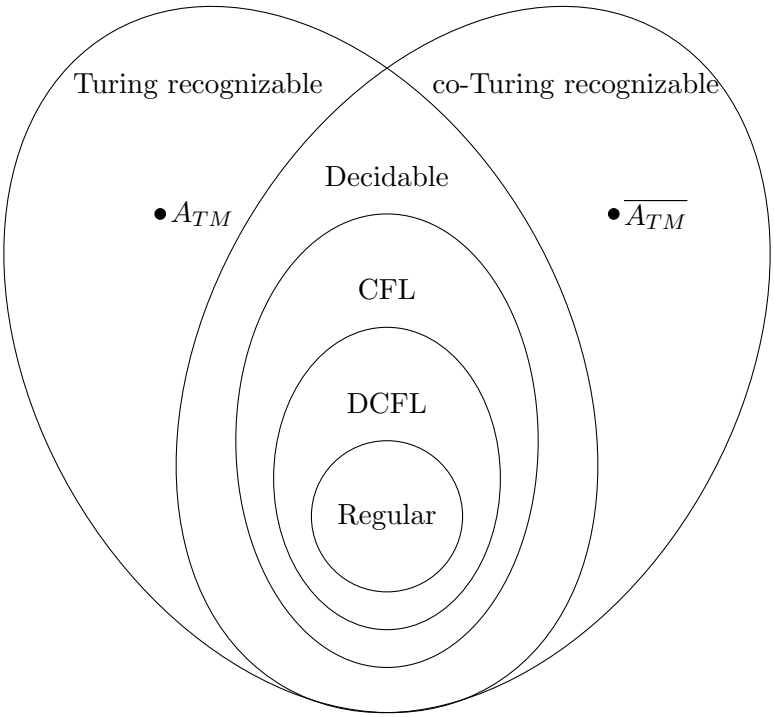
Proposition 3. *If a language L is undecidable and Turing recognizable then $\overline{L}$ is not Turing recognizable.*

3 Relationship Summary of Various Classes

Define

$$\text{co-Turing recognizable} = \{L \mid \overline{L} \text{ is Turing recognizable}\}.$$

The relation between the various classes that we have seen so far can be summarized as follows.



4 Reducibility

- Converting one problem to another.
- Examples:

(a) going around in a city reduces to ability to read a map

(b) getting good grades reduces to solving problems

(c) finding max/min reduces to sorting

(d) multiplication reduces to addition (doing it several times)

4.1 Computable Functions

Henceforth, we will also discuss TMs that compute a general function as opposed to outputting just *accept* or *reject*. We will assume that a TM has three tapes, namely

- a read-only *input tape*,
- a read-write *work tape*, and
- a write-only *output tape*.

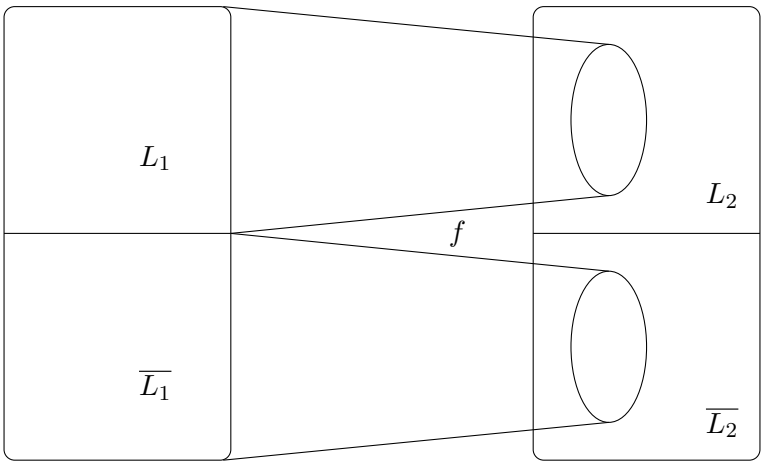
Since the output tape is write-only, the tape head of the output tape is restricted to move only from left to right. There is a special *halting state* in the TM. On any input, as soon as the TM enters the halting state, the computation of the TM stops and whatever string is written on the output tape, is the output of the TM on that particular input. For a TM M and on an input x, we will denote $M(x)$ to be the output of M on x.

Definition 4.1. A function $f : \Sigma^* \to \Sigma^*$ is said to be *computable* if there exists a TM M with output tape, such that for all $x \in \Sigma^*$, $f(x) = M(x)$.

Definition 4.2. We say that a language L_1 reduces to a language L_2 if there exists a computable function f, such that for all $x \in \Sigma^*$,

$$x \in L_1 \iff f(x) \in L_2.$$

This is denoted as $L_1 \leq_m L_2$.



- Note that f is a function, hence *every* string gets mapped to some string. But every string in L_2 or $\overline{L_2}$ need not have a pre-image.
- The m in the notation of reduction stands for "many-to-one".

- If $L_1 \leq_m L_2$ then intuitively it means that L_2 is at least as hard as L_1. In other words, a solution to L_2 would give a solution to L_1.
- A halting TM is also known as a *decider*.

Theorem 4. *If $L_1 \leq_m L_2$ and L_2 is decidable then L_1 is decidable.*

Proof. Let f be the computable function such that for all $x \in \Sigma^*$, $x \in L_1 \iff f(x) \in L_2$. Let M be a decider for L_2. We can then construct a decider for L_1 as follows.
Given $x \in \Sigma^*$, first compute $f(x)$. Simulate M on $f(x)$. If M accepts then *accept* else *reject*. □

Corollary 5. *Let L_1 and L_2 be two languages such that $L_1 \leq_m L_2$.*

(a) If L_1 is undecidable then L_2 is undecidable.

(b) If L_1 is not Turing recognizable then L_2 is not Turing recognizable.

5 Examples

1. Consider the language

$$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}.$$

Claim 6. $\overline{A_{TM}} \leq_m E_{TM}$.

We will construct a computable function f that takes as input $\langle M, w \rangle$ and produces an output $\langle M' \rangle$ such that M does not accept $w \iff L(M') = \emptyset$.

The reduction function f

Input: $\langle M, w \rangle$

(a) Construct a TM M' that on input x does the following (note that M' has description of M and w hardcoded into its description):
   i. If $x \neq w$ then *reject*
   ii. Else simulate M on w. If M accepts then *accept* and if M rejects then *reject*.

Output: $\langle M' \rangle$

Proof of correctness

Now,

$$\begin{aligned} M \text{ does not accept } w &\implies M' \text{ does not accept any input} \implies L(M') = \emptyset \\ M \text{ accepts } w &\implies M' \text{ accepts only the string } w \implies L(M') = \{w\} \end{aligned}$$

Therefore, M does not accept $w \iff L(M') = \emptyset$ and hence $\overline{A_{TM}} \leq_m E_{TM}$. This proves that E_{TM} is not Turing recognizable.

2. Consider the language

$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}.$$

Claim 7. $E_{TM} \leq_m EQ_{TM}$.

We will construct a computable function f that takes as input $\langle M\rangle$ and produces an output $\langle M_1, M_2\rangle$ such that $L(M) = \emptyset \iff L(M_1) = L(M_2)$.

The reduction function f

Input: $\langle M\rangle$

(a) Set $M_1 := M$.

(b) Construct a TM M_2 that rejects all inputs (that is $L(M_2) = \emptyset$).

Output: $\langle M_1, M_2\rangle$

Proof of correctness

Now,

$$L(M) = \emptyset \iff L(M_1) = \emptyset \iff L(M_1) = L(M_2)$$

Therefore, $L(M) = \emptyset \iff L(M_1) = L(M_2)$ and hence $E_{TM} \leq_m EQ_{TM}$. This proves that EQ_{TM} is not Turing recognizable.

3. Consider the language

$$REG_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}.$$

Claim 8. $\overline{A_{TM}} \leq_m REG_{TM}$.

We will construct a computable function f that takes as input $\langle M, w\rangle$ and produces an output $\langle M'\rangle$ such that M does not accept $w \iff L(M')$ is regular.

The reduction function f

Input: $\langle M, w\rangle$

(a) Construct a TM M' that on input x does the following (note that M' has description of M and w hardcoded into its description):

i. Simulate M on w.

ii. If M accepts w then check if x is of the form 0^n1^n or not. If so then *accept* else *reject*.

iii. If M rejects w then *reject*.

Output: $\langle M'\rangle$

Proof of correctness

Now,

$$\begin{aligned} M \text{ does not accept } w &\implies M' \text{ does not accept any input} \implies L(M') = \emptyset \implies L(M') \text{ is regular} \\ M \text{ accepts } w &\implies L(M') = \{0^n 1^n \mid n \geq 0\} \implies L(M') \text{ is not regular} \end{aligned}$$

Therefore, M does not accept $w \iff L(M')$ is regular. Hence $\overline{A_{TM}} \leq_m REG_{TM}$. This proves that REG_{TM} is not Turing recognizable.

Similar approach can be used to show that checking whether the language of a TM is a CFL or not is undecidable.

4. Consider the language

$$CON_{TM} = \{\langle M_1, M_2, M_3\rangle \mid M_1, M_2, M_3 \text{ are TMs and } L(M_1) = L(M_2) \cdot L(M_2)\}.$$

Claim 9. $EQ_{TM} \leq_m CON_{TM}$.

We will construct a computable function f that takes as input $\langle N_1, N_2\rangle$ and produces an output $\langle M_1, M_2, M_3\rangle$ such that $L(N_1) = L(N_2) \iff L(M_1) = L(M_2) \cdot L(M_3)$.

The reduction function f

Input: $\langle N_1, N_2\rangle$

(a) Set $M_1 := N_1$ and $M_2 := N_2$.

(b) Construct a TM M_3 that accepts the string ϵ only and rejects every other string.

Output: $\langle M_1, M_2, M_3\rangle$

Proof of correctness

Now,

$$L(N_1) = L(N_2) \iff L(N_1) = L(N_2) \cdot \{\epsilon\} \iff L(M_1) = L(M_2) \cdot L(M_3)$$

Therefore, $L(N_1) = L(N_2) \iff L(M_1) = L(M_2) \cdot L(M_3)$ and hence $EQ_{TM} \leq_m CON_{TM}$. This proves that CON_{TM} is not Turing recognizable.

Exercise 3. If $L_1 \leq_m L_2$ then $\overline{L_1} \leq_m \overline{L_2}$.

6 Rice's Theorem

- A *property* of a language is a function,

$$P : \{\text{set of all languages}\} \longrightarrow \{0, 1\}.$$

- For a language L, if $P(L) = 1$ then we say that L *satisfies* P and if $P(L) = 0$ then we say that L *does not satisfy* P.

- Examples of properties of languages:

(i) the language contains the string 01001

(ii) the language is empty

(iii) the language has exactly 340 strings

- A property P is said to be a *non-trivial* property of languages of Turing machines if there exists two TMs M_1 and M_2 such that $P(L(M_1)) = 0$ and $P(L(M_2)) = 1$.

Theorem 10 (Rice's Theorem)**.** *Let P be a non-trivial property of languages of TMs. Then the language*

$$L_P = \{\langle M \rangle \mid L(M) \text{ satisfies } P\}$$

is undecidable.

Proof. We will show that $A_{TM} \leq_m L_P$.

Case 1 $P(\emptyset) = 0$

Since P is a non-trivial property, there exists a TM N such that $P(L(N)) = 1$. We will construct a computable function f that takes as input $\langle M, w \rangle$ and produces an output $\langle M' \rangle$ such that M accepts $w \iff L(M')$ satisfies P.

The reduction function f

Input: $\langle M, w \rangle$

1. Construct a TM M' that on input x does the following (note that M' has description of M and w hardcoded into its description):
   (a) Simulate M on w.
   (b) If M accepts w then simulate N on x. If N accepts then *accept* and if N rejects then *reject*.
   (c) If M rejects w then *reject*.

Output: $\langle M' \rangle$

Proof of correctness

Now,

$$\begin{aligned} M \text{ accepts } w &\implies L(M') = L(N) \implies L(M') \text{ satifies } P \\ M \text{ does not accept } w &\implies L(M') = \emptyset \implies L(M') \text{ does not satisfy } P \end{aligned}$$

Therefore $A_{TM} \leq L_P$.

Case 2 $P(\emptyset) = 1$

We can reduce this to Case 1. The complement property $\overline{P}$ is also a non-trivial property of languages of TMs. Since $P(\emptyset) = 1$ therefore $\overline{P}(\emptyset) = 0$. By Case 1 we have $A_{TM} \leq_m L_{\overline{P}}$. Hence $\overline{A_{TM}} \leq_m \overline{L_{\overline{P}}}$ (see Exercise 3). But $\overline{L_{\overline{P}}} = L_P$. Therefore $\overline{A_{TM}} \leq L_P$.

Therefore L_P is undecidable. □

Note 2. Usually in the exam you will be asked to prove that a language is undecidable without using Rice's Theorem. So you must know how to proceed with the argument of such proofs.