Lecture Note 5
Decidability and Reducibility of Computational Problems
Chapter 4. Decidability

A part of Part Two: Computability Theory consisting of

- Chapter 3. The Church-Turing Thesis
- Chapter 4. Decidability
- Chapter 5. Reducibility
- Other advanced topics
Figure 1: The Chomsky Hierarchy
4.1 Decidable Languages

We will discuss computational problems (languages) concerning computational systems we have learned so far.

We will use a lot of “simulation techniques”.
Decidable problems concerning regular language

*Accepting problem* for DFAs: testing if a given DFA accepts a given string

\[ A_{DFA} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts string } w \} \]

**Theorem 4.1** \( A_{DFA} \) is a decidable language.
Theorem 4.1 $A_{DFA}$ is a decidable language.

**proof idea**

Use a TM, on input $\langle B, w \rangle$, to simulate $B$ on $w$.
- keep track of $B$'s current state and position on $w$.
- the TM accepts $\langle B, w \rangle$ iff $B$ accepts $w$

Details:
- how is $B$ encoded?
- how to keep track of $B$ states?
- how to keep track of positions on $w$?
Similarly

\[ A_{NFA} = \{ \langle B, w \rangle \mid \text{B is an NFA that accepts string } w \} \]

**Theorem 4.2** \( A_{NFA} \) is a decidable language.

**proof ideas**

1. Use an NTM to simulate NFA \( B \) on \( w \),
   - accepts \( \langle B, w \rangle \) iff \( B \) accepts \( w \).

2. Alternatively, convert \( B \) an equivalent DFA \( B' \),
   - then run the TM \( M_1 \) for Theorem 4.1 on \( w \)
   - accepts \( \langle B, w \rangle \) iff \( M_1 \) accepts \( \langle B', w \rangle \).
\[ A_{\text{REX}} = \{(R, w) \mid \text{regular expression } R \text{ generates } w\} \]

**Theorem 4.3** $A_{\text{REX}}$ is a decidable language.

**proof idea?**
Testing if a finite automaton accepts the empty language (rejecting all strings)

\[ E_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \phi \} \]

**Theorem 4.4** \( E_{DFA} \) is a decidable language.

**proof idea**

To determine *if there is path from the initial state to a accept state.*
Testing if two DFAs are equivalent

\[ EQ_{DFA} = \{ \langle A, B \rangle \mid A, B \text{ are DFAs and } L(A) = L(B) \} \]

**Theorem 4.5** \( EQ_{DFA} \) is a decidable language.

**proof idea:**
To relate testing \( L(A) = L(B) \) to a empty language testing. How?
Differences $L(A) - L(B)$ and $L(B) - L(A)$ are both empty if and only if $L(A) = L(B)$

$L(A) - L(B) = L(A) \cap \overline{L(B)}$

So there is a DFA $C$, constructed from $A$ and $B$ such that $L(C) = L(A) - L(B)$

$L(B) - L(A) = L(B) \cap \overline{L(A)}$

So there is a DFA $C'$, constructed from $A$ and $B$ such that $L(C') = L(B) - L(A)$

Construct a TM to check emptiness of $L(C)$ and $L(C')$ according to the proof of Theorem 4.4.
Decidable problems concerning context-free languages

\[ A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \]

**Theorem 4.7** \( A_{CFG} \) is a decidable language.

**proof idea:**

Use \( G \) to try all possible derivations to see if any of them derive \( w \).
- However, the process may not stop;
- Would only prove Turing recognizable for \( A_{CFG} \).
To use Chomsky normal form
\[(S_0 \rightarrow \epsilon, X \rightarrow YZ, X \rightarrow a)\]

(1) can check if \(w = \epsilon\)

(2) let \(|w| = l\), assume there is a derivation,
   - how many steps needed to derive \(w\)?
   - each use of rule \(X \rightarrow YZ\) generates at least one symbol
   - like \(X \Rightarrow YZ \Rightarrow aZ\), two steps

Derivation of one symbol needs two steps except the last one.
So totally \(2l - 1\) steps

Try all derivations of \(2l - 1\) steps.
   (how many of them? \(|R|^{2l-1}\))
Also we have the problem of testing if a CFG generates the empty language.

\[ E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \phi \} \]

**Theorem 4.8** \( E_{CFG} \) is a decidable language.

**proof idea:**

Not work: enumerating all strings \( w \) and checking if the given grammar generates it.

Check if the start variable derives a terminal string?
  - but the start variables depends on other variables

Check if each variable derives a terminal string?
Theorem 4.8 $E_{CFG}$ is a decidable language.

Proof:
On input $\langle G \rangle$, encoding of $G$,
- mark all terminal symbols in $G$,
- repeat until no new variables get marked:
  - mark any variable $A$ if $A \rightarrow \alpha$ is a rule
    and all symbols in $\alpha$ has been marked.
- if the start variable is not marked, accept; otherwise reject.
We would hope the following language is also decidable, but it is not.

$$EQ_{CFG} = \{ (G, H) \mid G, H \text{ are CFGs and } L(G) = L(H) \}$$

The technique used to prove Theorem 4.5 is not applicable to CFLs.

- There language intersection and complement are used to create a difference language
- But the class of CFLs are not closed under intersection or complement
(1) Why is the class of CFLs not closed under intersection?

\[
\{a^n b^n c^n | n \geq 0\} = \{a^n b^n c^m | m, n \geq 0\} \cap \{a^m b^n c^n | m, n \geq 0\}
\]

non-CFL = CFL, contradiction!

(2) Why is the class of CFLs not closed under complement?

Otherwise,

\[
\{a^n b^n c^n | n \geq 0\} = \overline{\{a^n b^n c^m | m, n \geq 0\} \cap \{a^m b^n c^n | m, n \geq 0\}}
\]

\[
\{a^n b^n c^n | n \geq 0\} = \overline{\{a^n b^n c^m | m, n \geq 0\} \cup \{a^m b^n c^n | m, n \geq 0\}}
\]

non-CFL = CFL, contradiction!
Theorem 4.9 Every CFL is decidable.

proof idea:

- Let \( G \) be the CFG for the language under consideration.
- A TM can be used to simulate \( G \) on input \( w \), as it is done in Theorem 4.7.

Figure 4.10 The relationship among classes of languages

regular \( \subset \) context-free \( \subset \) decidable \( \subset \) Turing-recognizable
4.2 The Halting Problem

Similar to the DFA and CFG settings, we define

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

However, unlike $A_{DFA}$ and $A_{CFG}$, $A_{TM}$ is not decidable.

But $A_{TM}$ is Turing-recognizable.

Proof by constructing an ‘universal’ Turing machine to simulate $M$ on $w$. 
The Diagonalization Method [Georg Cantor, 1873]
For two infinite sets, how to tell which one is larger?
- how to compare the two infinite sets?
- pairing elements in the two sets.

Definition 4.12
Let $A$ and $B$ are two sets and $f : A \to B$.

1. $f$ is one-to-one if $f(a) \neq f(b)$ whenever $a \neq b$.
2. $f$ is onto if for every $b \in B$, there is an $a$, $f(a) = b$, and
3. $f$ is correspondence if it is both one-to-one and onto.

Two sets are the same size if there is a correspondence between them.
Example 4.13.
Natural number set $\mathcal{N}$ and even number set $\mathcal{E}$ are the same size.

**Definition 4.14**
A set is **countable** if either it is finite or it has the same size as $\mathcal{N}$.

Figure 4.16
A correspondence between $\mathcal{N}$ and the rational number set $\mathbb{Q}$. 
\( \mathcal{R} \), the real number set is actually much larger!

**Theorem 4.17** \( \mathcal{R} \) is not countable.

**proof idea**

- Assume that there is a correspondence between \( \mathcal{R} \) and \( \mathcal{N} \).
- Construct a real number \( x \) that does not have a natural number to pair.
  
  (1) align all real numbers based on their decimal points,
  (2) let \( x = 0.x_1x_2\ldots \) such that
  \( x_i \) is different from the \( i \)th digit after the decimal point
  of the real number \( r_i \) that corresponds to the natural \( i \).

- Then \( x \) is different from all the real numbers
  Yet it has NO natural number to correspond. Contradict!
Using almost the same idea to prove

**Theorem 4.11** $A_{TM}$ is not decidable.

**proof idea:**

- Assume TM $H$ that can decides if $M$ accepts $w$
  for any given $M$ and $w$. That is

  \[
  H(\langle M, w \rangle) = \begin{cases} 
  \text{accepts} & M \text{ accepts } w \\
  \text{reject} & M \text{ rejects } w 
  \end{cases}
  \]  

- Built another TM $D$ that **checks if a given TM accepts itself**

  On input $\langle M \rangle$, encoding a TM $M$,
  - $D$ simulates $H$ on $\langle M, w \rangle$, where $w = M$
  - $D$ accepts $\langle M \rangle$ iff $H$ rejects $\langle M, M \rangle$. That is

  \[
  D(\langle M \rangle) = \begin{cases} 
  \text{accepts} & H \text{ rejects } \langle M, M \rangle \\
  \text{reject} & H \text{ accepts } \langle M, M \rangle 
  \end{cases}
  \]
(3) Consider input \(\langle D \rangle\) to TM \(D\),

Will \(D\) accepts \(\langle D \rangle\)?

**Scenario 1:**

\(D\) accepts \(D\). That is \(H\) rejects \(\langle D, D \rangle\) by formula (2).

However, by formula (1) \(D\) should rejects \(D\). **Contradicts!**

**Scenario 2:**

\(D\) rejects \(D\). That is \(H\) accepts \(\langle D, D \rangle\) by formula (2).

However, by formula (1) \(D\) should accept \(D\). **Contradicts!**

Called *paradox*. So neither \(H\) nor \(D\) can exist.

**Where is the diagonalization?**

Figures 4.19, 4.20, 4.21.
Logical paradox exists in some statements.
e.g., A barber said he serves anyone who does not cut his own hair.
Question: does the barber cuts his own hair?
   (1) If yes, then he is one of those who cuts his own hair.
       Then he should not serve himself. Contradics.
   (2) If not, then is one of those who does not cut his own
       hair. Then he should himself. Contradics.
Russell’s paradox (Russell’s antinomy):
Let $R$ be the set of all sets that are not members of themselves, i.e., assume

$$R = \{x \mid x \notin x\}$$

Question: $R \in R$?

Both the set of people who the Barber serves and the Turing machine $D$ constructed for Theorem 4.14 can be thought of such a set $R$. 
So we proved that some languages are not decidable, e.g., $A_{TM}$.

But $A_{TM}$ is Turing-recognizable.

Are there languages that are NOT Turing-recognizable?

**Lemma** If language $L$ is Turing-recognizable and its complement $\overline{L}$ is also Turing-recognizable, then $L$ is decidable.

**Corollary 4.23:** $A_{TM}$ is Turing-recognizable but not decidable, so its complement $\overline{A_{TM}}$ is NOT Turing-recognizable.
**Lemma** If language $L$ is Turing-recognizable and its complement $\overline{L}$ is also Turing-recognizable, then $L$ is decidable.

**proof idea.**
Simulate both TMs ($A$ for $L$, $B$ for $\overline{L}$) on input $w$,
accept $w$ if $A$ accepts $w$; reject $w$ if $B$ accepts $w$.

**Lemma** If $L$ is decidable, then both $L$ and $\overline{L}$ are at least Turing-recognizable.

**Theorem 4.22** A language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable.
(co-Turing-recognizable means the complement is Turing-recognizable).
Chapter 5. Reducibility

A part of Part Two: Computability Theory consisting of

Chapter 3. The Church-Turing Thesis
Chapter 4. Decidability
Chapter 5. Reducibility
Other advanced topics
We will investigate how to use solutions for one problem to solve another problem.

This will allow us to show more problems that are undecidable.

- Two problems $A$ and $B$. If $A$ is reduced to $B$, a solution for $B$ can be used to solve problem $A$
  - That is, if you have solutions for $B$, you have solutions for $A$.
  - In other words, if there is no solution for $A$, then there is no solution for $B$.
  - I.e., $A$ is NOT more difficult than $B$
  - But $B$ is at least as difficult as $A$ (may be more difficult than $A$).
5.1 Undecidable Problems from Language Theory

\[ \text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \} \]

The problem is as hard as \( A_{TM} \), undecidable.

**Theorem 5.1** \( \text{HALT}_{TM} \) is undecidable.

**proof idea:**

(1) assume the opposite, \( \text{HALT}_{TM} \) is decidable, admitting a TM \( R \).

(2) construct a TM to decide \( A_{TM} \), on input \( \langle M, w \rangle \)

- simulate \( R \) on \( \langle M, w \rangle \),
- if \( R \) rejects, reject,
- if \( R \) accepts, simulate \( M \) on \( w \) until it halts
  - accept if \( M \) accepts \( w \); reject if \( M \) rejects \( w \).

This is a decider for \( A_{TM} \), contradicts!
Testing the emptiness:

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a Turing machine and } L(M) = \phi \} \]

**Theorem 5.2** \( E_{TM} \) is undecidable.

**proof idea:**

We want to show that if \( E_{TM} \) is decidable then \( A_{TM} \) is also decidable, leading to a contradiction. We assume there is an algorithm \( R \) for \( E_{TM} \).

- Input \( \langle M, w \rangle \) for the \( A_{TM} \), run \( R \) on \( M \) can only know if \( L(M) = \phi \) or not.

- **How to construct a TM \( M_1 \) such that**

  \[ L(M_1 = \phi \text{ if and only if } \langle M, w \rangle \in A_{TM} \text{ (i.e., } M \text{ accepts } w) \]
On the input $\langle M, w \rangle$:

(1) Construct $M_1$ such that it will reject all strings $x \neq w$
and the rest construction is the same as $M$.
(so $M_1$ accepts at most one string $w$.)

(2) Now run $R$ on $M_1$ to see if $L(M_1) = \phi$.
   reject the input $\langle M, w \rangle$ iff $R$ accepts $M_1$.

Step (1) is finite. Step (2) can halt since $R$ can halt.

So this results in a decider for $A_{TM}$. Contradicts!
What do you conclude from the previous proof?

$A_{TM}$ reduced to $E_{TM}$

Given input $\langle M, w \rangle$ construct TM $M_1$

$M$ accepts $w \implies L(M_1) = \{w\}$

$M$ rejects $w \implies L(M_1) = \emptyset$

So if there is a decider $R$ solves the second question, the first question can be answered as well.

NOTE: the reduction does not run $M$ on $w$. 
Testing the regularity:

\[ \text{REGULAR}_{TM} = \{(M) | M \text{ is a Turing machine} \& L(M) \text{ is regular}\} \]

**Theorem 5.3** \( \text{REGULAR}_{TM} \) is undecidable.

**proof idea**: Use the previous proof idea:

- Given input \( \langle M, w \rangle \)
- construct TM \( M_2 \)
- \( M \) accepts \( w \) \( \implies \) \( L(M_2) = \{0^n1^n | n \geq 0\} \cup \Sigma^* \)
- \( M \) rejects \( w \) \( \implies \) \( L(M_2) = \{0^n1^n | n \geq 0\} \)
Assume a decider $R$ for $REGULAT_{TM}$.

On the input $\langle M, w \rangle$:

1. Construct $M_2$ so it accepts any input $x$ in the form of $0^n1^n$ and if the input is not in that form, it follows execution of $M$ on $w$.
   (NOTE: $M_2$ accepts $\Sigma^*$ in the case $M$ accepts $w$.)
   (NOTE: $M_2$ accepts $\{0^n1^n | n \geq 0\}$ in the case $M$ rejects $w$)

2. Now run $R$ on $M_2$ to see if $L(M_2)$ is regular;
   accept the input $\langle M, w \rangle$ iff $R$ accepts $M_2$.

Step (1) is finite. Step (2) can halt since $R$ can halt.

So this results in a decider for $A_{TM}$. Contradicts!
Testing the equality of two languages recognized by TMs.

\[ E_{QM} = \{ (M_1, M_2) \mid L(M_1) = L(M_2) \} \]

**Theorem 5.4** \( E_{QM} \) is undecidable.

**proof idea?**
3.2 Mapping Reducibility

In previously used reductions:

\[ \text{language } A \implies \text{language } B \]

mapping an instance of \( A \) to some instance of \( B \)

To use the reduction to prove property for \( A \)

\[ \text{answers for } A \iff \text{answers for } B. \]

- no rules set for what \( \implies \) should be
- no rules set for what \( \iff \) should be

a reduction can be used as long as it works.
A refined way to define reduction

the reduction needs to be done by an algorithm

the answers for $B$ can be interpreted in finite steps
Definition 5.17

A function \( f : \Sigma^* \rightarrow \Sigma^* \) is a **computable function** if some TM \( M \), on every input \( w \), halts with just \( f(w) \) on its tape.

Picture it:

Note: some inputs \( w \) may not have defined \( f(w) \)

Examples:

1. \( f(x, y) = x + y \)
   
   based on unary representation
   based on binary representation

2. \( f(w) = w^{-1} \), reversing strings
Formal definition of mapping reduction

**Definition 5.20** language $A$ is **mapping reducible** to language $B$, denoted as $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, such that for every $w$

$$w \in A \text{ if and only if } f(w) \in B$$

The function $f$ is called the **reduction** from $A$ to $B$.

Picture it (figure 5.21, page 207)

A mapping reduction transforms each membership testing for $A$ to some membership testing for $B$. 
Theorem 5.22
If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

Proof:
Assume $M$ to be the decidable for $B$ and $f$ be the mapping reduction. Construct a decidable $N$ for $A$ as follows.

$N = \text{“On input } w \text{”}$
1. Compute $f(w)$.
2. Run $M$ on input $f(w)$ and
   accepts (rejects) $w$ iff $M$ accepts (rejects) $f(w)$ ”
Is the other way also true?

**Corollary 5.23**

If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

Also similar to Theorem 5.22

**Theorem 5.28**

If $A \leq_m B$ and $B$ is Turing-recognizable, then $A$ is Turing-recognizable.
Now we re-visit some old proofs using the concept of mapping reduction.

Example 5.26. Proof for that $EQ_{TM}$ is not decidable
- by mapping reduction from $E_{TM}$, $E_{TM} \leq EQ_{TM}$, vis $f$
- $f$ is such: $f(\langle M \rangle) = \langle M, M_1 \rangle$, where $M_1$ is a Turing machine accepting no strings.

$\langle M \rangle \in E_{TM}$ if and only if $\langle M, M_1 \rangle \in EQ_{TM}$
Example: Proof for that $\textit{REGULAR}_{TM}$ is not decidable

- by mapping reduction from $A_{TM}$, $A_{TM} \leq \textit{REGULAR}_{TM}$,

- $f$ is such: $f(⟨M,w⟩) = ⟨M_1⟩$, where

  $M_1$ is a Turing machine constructed as follows

  (a) it accepts all strings of format $0^n1^n$,
  (b) it accepts all other strings if $M$ accepts $w$

  $⟨M,w⟩ \in A_{TM}$ if and only if $⟨M_1⟩ \in \textit{REGULAR}_{TM}$

That is

$⟨M,w⟩ \in A_{TM} \implies L(M_1) = \Sigma^*$

$⟨M,w⟩ \notin A_{TM} \implies L(M_1) = \{0^n1^n|n \geq 0\}$
Theorem 5.28
If $A \leq_m B$ and $B$ is Turing-recognizable, then $A$ is Turing-recognizable.

Corollary 5.29
If $A \leq_m B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.
There are problems that are not Turing-recognizable. For example, $A_{TM}$ is Turing-recognizable but not decidable. So

- $\overline{A_{TM}}$ is not Turing-recognizable, or equivalently,
- $A_{TM}$ is NOT co-Turing-recognizable.

**Theorem 5.30**

$EQ_{TM}$ is neither Turing-recognizable nor co-Turing-recognizable.

**proof idea**

- Note that $A \leq_m B$ is the same as $\overline{A} \leq_m \overline{B}$
- To prove $EQ_{TM}$ is not Turing recognizable, we construct $\overline{A_{TM}} \leq_m EQ_{TM}$, or equivalently, $A_{TM} \leq_m EQ_{TM}$
proof idea (cont.)

we construct \( \overline{A_{TM}} \leq_m EQ_{TM} \) via the mapping reduction function:

\[ f : f(\langle M, w \rangle) = \langle M_1, M_2 \rangle, \]

where

\( M_1 = \) "rejects all strings".
\( M_2 = \) "accepts all strings if \( M \) accepts \( w \) [How ?].

That is, \( M \) rejects \( w \) iff \( L(M_1) = L(M_2) \).

This proves that \( EQ_{TM} \) is not-Turing-recognizable.
To prove $\overline{EQ_{TM}}$ is not-Turing-recognizable, we construct $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ via the mapping reduction function:

$f : f(\langle M, w \rangle) = \langle M_1, M_2 \rangle$, where

$M_1 =$ "accept all strings".

$M_2 =$ “accepts all strings if $M$ accepts $w$.

That is, $M$ rejects $w$ iff $L(M_1) \neq L(M_2)$. 
Review:

Chapter 1. Regular Languages
- DFA, NFA, regular expressions,
- equivalence of NFAs and DFAs
- equivalence of FAs and regular expressions
- closure under the regular operations
- non-regular languages, pumping lemma
Chapter 2. Context-Free Languages

- CFG, ambiguity, Chomsky normal form
- PDA
- equivalence of CFGs and PDAs
- non-CFL, pumping lemma
- CFLs are not closed under intersection and complement
Chapter 3.
- TM, configuration
- variants, nondeterministic TM
- definition of algorithms, Turing-recognizability

Chapter 4.
- decidable languages, examples
- proving decidability
- diagonalization, proof ATM is undecidable

Chapter 5.
- proving undecidability
- mapping reduction, computable functions
The End of Lecture Notes for CSCI 2670 Fall 2012

*Please email typos and other errors to cai@cs.uga.edu.*