Theory of Languages and Automata

Chapter 4- Decidability

Sharif University of Technology

Decidable Languages

- Let
  $A_{DFA} = \{\langle B, w\rangle \mid B \mathit{\ is\ a\ DFA\ that\ accepts\ input\ string\ } w\}.$
- Theorem 4.1
  - ✓ A_{DFA} is decidable.

Decidable Languages (Proof Idea)

- We simply need to present a TM M that decides A_{DFA}.

- M = "On input $\langle B, w\rangle$, where B is a DFA and w is a string:
  1. Simulate B on input w.
  2. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

Decidable Languages (cont.)

- Let
  $A_{DFA} = \{\langle B, w \rangle \mid B \, is \, a \, NFA \, that \, accepts \, input \, string \, w\}.$
- Theorem 4.2
  - A_{DFA} is decidable.

Decidable Languages (Proof)

- We design TM N that decides A_{DFA}.

- N = "On input $\langle B, w\rangle$, where B is a NFA and w is a string:
  1. Convert NFA B to an equivalent DFA C, using the procedure for this conversion given in Theorem 1.39.
  2. Run TM M from the Theorem 4.1 on input $\langle C, w\rangle$.
  3. If M accepts, *accept;* otherwise, *reject.*"

Decidable Languages (cont.)

- Let

$$A_{REX} = \{\langle R, w \rangle \mid R \textit{ is a regular expression that generates string } w\}.$$

- Theorem 4.3
  - ✓ A_{REX} is decidable.

Decidable Languages (Proof)

- The following TM P decides A_{REX}.

- $P =$ "On input $\langle R, w \rangle$, where R is a regular expression and w is a string:
  1. Convert regular expression R to an equivalent NFA A, using the procedure for this conversion given in Theorem 1.54.
  2. Run TM N on input $\langle A, w \rangle$.
  3. If N accepts, *accept;* in N rejects, *reject.*"

Decidable Languages (cont.)

- Let

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

- Theorem 4.3
  - ✓ E_{DFA} is decidable.

Proof

- The following TM T decides E_{DFA}:
- T = "On input $\langle A \rangle$ where A is a DFA:
  1. Mark the start state of A.
  2. Repeat until no new states get marked:
  3. Mark any state has a transition coming into it from any state that is already marked.
  4. If no accept state is marked, *accept*; otherwise, *reject*."

Proof

- The following TM T decides A_{DFA}:
- T = "On input $\langle A \rangle$ where A is a DFA:
  1. Mark the start state of A.
  2. Repeat until no new states get marked:
  3. Mark any state has a transition coming into it from any state that is already marked.
  4. If no accept state is marked, *accept*; otherwise, *reject*."

Decidable Languages (cont.)

- Let:
  $EQ_{DFA} = \{\langle A, B\rangle | \ A \ and \ B \ are \ DFAs \ and \ L(A) = L(B)\}$.

- **Theorem 4.5:** EQ_{DFA} is decidable.

Proof

- Construct DFA C such that:

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

- TM F decides EQ_{DFA} as follows:
- F = “On input ⟨A, B⟩, where A and B are DFAs:
  1. Construct DFA C as described.
  2. Run TM T from Theorem 4.4 on input ⟨C⟩.
  3. If T accepts, *accept*. If T rejects, *reject*.”

Decidable Languages (cont.)

- Let:
  $A_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates string } w\}$.

- **Theorem 4.5:** A_{CFG} is decidable.

Proof

- The TM S for A_{CFG} follows.
- S = "On 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 $2n - 1$ steps, where n is the length of w, except if $n = 0$, then instead list all derivations with 1 step.
  3. If any of these derivations generate w, *accept*; if not, *reject*."

Decidable Languages (cont.)

- Let:

$$E_{CFG} = \{\langle G \rangle |\ G\ is\ a\ CFG\ and\ L(G) = \emptyset\}.$$

- **Theorem 4.5:** E_{CFG} is decidable.

Proof

- TM R decides E_{CFG} as follows:
- R = "On input $\langle G \rangle$, where G is a CFG:
  1. Mark all terminal symbols in G.
  2. Repeat until no new variables get marked:
  3. Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \ldots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
  4. If the start variable is not marked, *accept*; otherwise, *reject*."

Decidable Languages (continued)

- **Theorem : Every** context-free language is decidable.

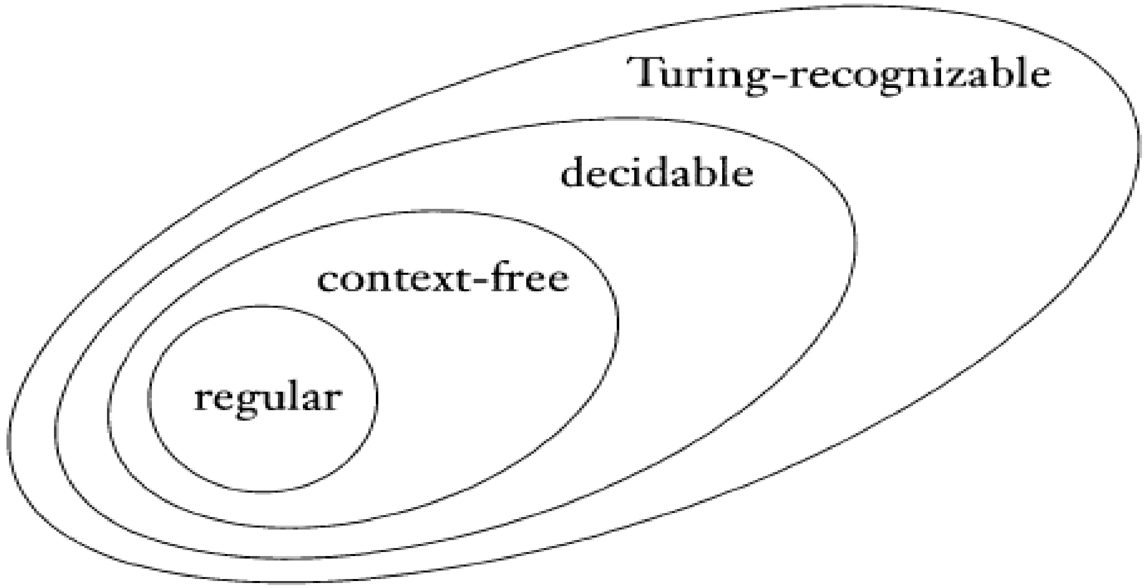
- **PROOF**

Let G be a CFG for A and design a TM M_G that decides A. We build a copy of G into M_G. It works as follows.

M_G = "On input w:

1. Run TM S on input *(G, w)*
2. If this machine accepts, *accept;* if it rejects, *reject."*

Relationship Between Four Classes of Languages



Undecidable Languages

- Let

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

- **Theorem 4.11:** Atm is undecidable.

Proof

- **PROOF** Let H be a decider for ATM. Then

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

Define D as follows

D = "On input $\langle M\rangle$, where M is a TM:
1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs; that is, if H accepts, *reject* and if H rejects, *accept*."

- Then

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

$$D(\langle D\rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \textit{reject} & \text{if } D \text{ accepts } \langle D\rangle. \end{cases}$$

- So

No matter what D does, it is forced to do the opposite, which is obviously a contradiction. Thus neither TM D nor TM H can exist.

Turing-Unrecognizable Languages

- **DEFINITION** A language is **co-Turing-recognizable** if it is the complement of a Turing-recognizable language.

Theorem

- **THEOREM:** A language is decidable if it is Turing-recognizable and co-Turing-recognizable.

- **PROOF** We have two directions to prove. First, if A is decidable, then both A and its complement A^C are Turing-recognizable.

  For the other direction, let M_1 be the recognizer for A and M_2 be the recognizer for A^C. The following Turing Machine M is a decider for A.

  M = "On input w:

  1. Run both M_1 and M_2 on input w in parallel.
  2. If M_1 accepts, *accept*; if M_2 accepts, *reject*."

Theorem

$\overline{A_{TM}}$ is not Turing-recognizable.

PROOF We know that ATM is Turing-recognizable. If $\overline{A_{TM}}$ also were Turing-recognizable, ATM would be decidable. Theorem 4.11 tells us that ATM is **not** decidable, so $\overline{A_{TM}}$ must not be Turing-recognizable.