CSCI 335 Theory of Computing

Lecture 15 – Decidability I: Decidable Languages

DATE : December 11 2022

Learning Outcomes

Learning Outcomes

By the end of this section you will be able to:

- Construct Turing machines for some decidable regular languages
- Construct Turing machines for some decidable context-free languages
- Get acquainted with examples of undecidable languages

Outline

- Decidable problems concerning regular languages
- Decidable problems concerning context-free languages
- Undecidable problems concerning context-free languages

Decidable problems concerning regular languages

Acceptance Problem for DFAs

Let $A_{\text{DFA}} = \{\langle B, w\rangle \mid B$ is a DFA and B accepts $w\}$

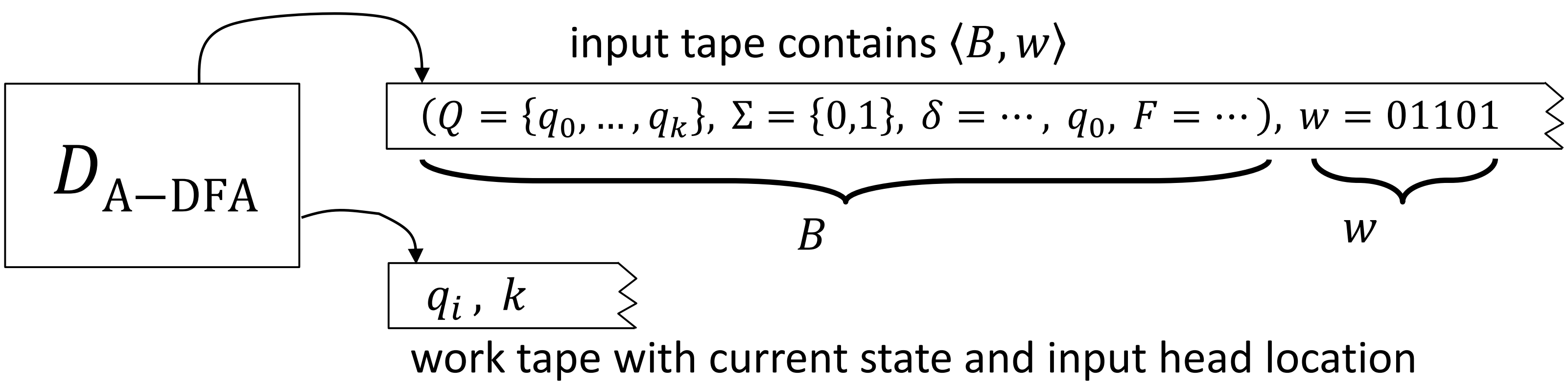
Theorem: A_{DFA} is decidable

Proof: Give TM $D_{\text{A-DFA}}$ that decides A_{DFA}.

$D_{\text{A-DFA}}$ = "On input s

1. Check that s has the form $\langle B, w\rangle$ where B is a DFA and w is a string; *reject* if not.
2. Simulate the computation of B on w.
3. If B ends in an accept state then *accept*. If not then *reject*."

Shorthand:
On input $\langle B, w\rangle$



Acceptance Problem for NFAs

Let $A_{\mathrm{NFA}} = \{\langle B, w\rangle \mid B$ is a NFA and B accepts $w\}$

Theorem: A_{NFA} is decidable

Proof: Give TM $D_{\mathrm{A-NFA}}$ that decides A_{NFA}.

$D_{\mathrm{A-NFA}} =$ "On input $\langle B, w\rangle$

1. Convert NFA B to equivalent DFA B'.
2. Run TM $D_{\mathrm{A-DFA}}$ on input $\langle B', w\rangle$. [Recall that $D_{\mathrm{A-DFA}}$ decides A_{DFA}]
3. *Accept* if $D_{\mathrm{A-DFA}}$ accepts.
   Reject if not."

New element: Use conversion construction and previously constructed TM as a subroutine.

Emptiness Problem for DFAs

Let $E_{\mathrm{DFA}} = \{\langle B\rangle |\ B$ is a DFA and $L(B) = \emptyset\}$

Theorem: E_{DFA} is decidable

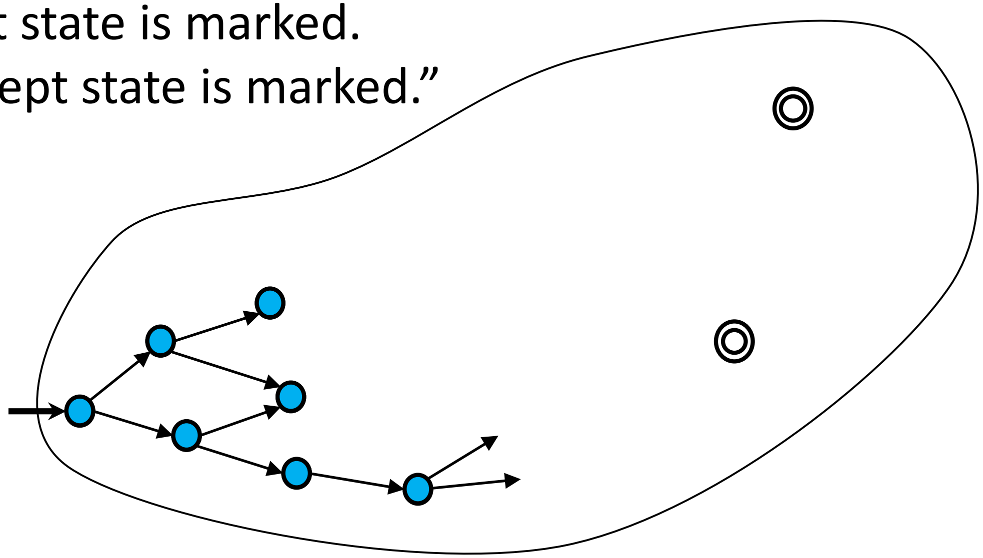
Proof: Give TM $D_{\mathrm{E-DFA}}$ that decides E_{DFA} .

$D_{\mathrm{E-DFA}} =$ "On input $\langle B\rangle$ [IDEA: Check for a path from start to accept.]

1. Mark start state.
2. Repeat until no new state is marked:

   Mark every state that has an incoming arrow from a previously marked state.

3. *Accept* if no accept state is marked.
   Reject if some accept state is marked."



Equivalence problem for DFAs

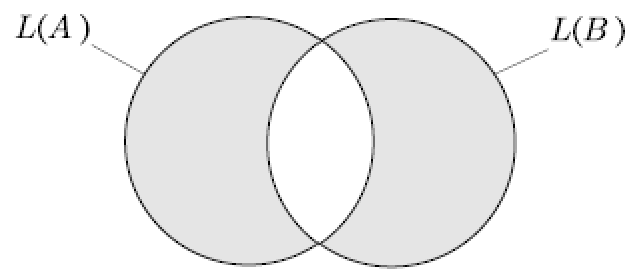
Let $EQ_{\mathrm{DFA}} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$

Theorem: EQ_{DFA} is decidable

Proof: Give TM $D_{\mathrm{EQ-DFA}}$ that decides EQ_{DFA} .

$D_{\mathrm{EQ-DFA}} =$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

1. Construct DFA C where $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$.
2. Run $D_{\mathrm{E-DFA}}$ on $\langle C\rangle$.
3. *Accept* if $D_{\mathrm{E-DFA}}$ accepts.
   Reject if $D_{\mathrm{E-DFA}}$ rejects."



Symmetric difference

Equivalence problem for DFAs

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

Theorem: EQ_{DFA} is decidable

Proof: Give TM $D_{\text{EQ-DFA}}$ that decides EQ_{DFA}.

$D_{\text{EQ-DFA}} =$ "On input $\langle A, B\rangle$ [IDEA: Make DFA C that accepts w where A and B disagree.]

1. Construct DFA C where $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$.
2. Run $D_{\text{E-DFA}}$ on $\langle C\rangle$.
3. *Accept* if $D_{\text{E-DFA}}$ accepts.
   Reject if $D_{\text{E-DFA}}$ rejects."

Check-in 7.1

Let $EQ_{\text{REX}} = \{\langle R_1, R_2\rangle | \ R_1$ and R_2 are regular expressions and $L(R_1) = L(R_2)\}$

Can we now conclude that EQ_{REX} is decidable?

a) Yes, it follows immediately from things we've already shown.

b) Yes, but it would take significant additional work.

c) No, intersection is not a regular operation.

Decidable problems concerning context-free languages

Acceptance Problem for CFGs

Recall Chomsky Normal Form (CNF) only allows rules:

A → BC

B → b

Lemma 1: Can convert every CFG into CNF.
Proof and construction in book.

Lemma 2: If H is in CNF and $w \in L(H)$ then every derivation of w has $2|w| - 1$ steps.
Proof: exercise.

Acceptance Problem for CFGs

Let $A_{\text{CFG}} = \{\langle G, w\rangle \mid G$ is a CFG and $w \in L(G)\}$

Theorem: A_{CFG} is decidable

Proof: Give TM $D_{\text{A-CFG}}$ that decides A_{CFG}.

$D_{\text{A-CFG}} =$ "On input $\langle G, w\rangle$

1. Convert G into CNF.
2. Try all derivations of length $2|w| - 1$.
3. *Accept* if any generate w.
   Reject if not.

Corollary: Every CFL is decidable.

Proof: Let A be a CFL, generated by CFG G.

Construct TM $M_G =$ "on input w

1. Run $D_{\text{A-CFG}}$ on $\langle G, w\rangle$.
2. *Accept* if $D_{\text{A-CFG}}$ accepts
   Reject if it rejects."

Check-in 7.2

Can we conclude that A_{PDA} is decidable?

a) Yes.

b) No, PDAs may be nondeterministic.

c) No, PDAs may not halt.

Emptiness Problem for CFGs

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

Theorem: E_{CFG} is decidable

Proof:

$D_{\text{E-CFG}}$ = "On input $\langle G \rangle$ [IDEA: work backwards from terminals]

1. Mark all occurrences of terminals in G.
2. Repeat until no new variables are marked
   Mark all occurrences of variable A if
   $\mathrm{A} \rightarrow \mathrm{B}_1 \mathrm{B}_2 \cdots \mathrm{B}_k$ is a rule and all B_i were already marked.
3. *Reject* if the start variable is marked.
   Accept if not."

Mark the terminals first. Then mark a variable if it is the antecedent in a rule whose all consequences are all marked.

$\mathrm{S} \rightarrow \mathrm{RTa}$
$\mathrm{R} \rightarrow \mathrm{Tb}$
$\mathrm{T} \rightarrow \mathrm{a}$

Example undecidable problems

Undecidable problems of context–free languages

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

Theorem: EQ_{CFG} is NOT decidable

Let $AMBIG_{\text{CFG}} = \{\langle G\rangle \mid G$ is an ambiguous CFG $\}$

Theorem: $AMBIG_{\text{CFG}}$ is NOT decidable

Check-in 7.3

Why can't we use the same technique we used to show EQ_{DFA} is decidable to show that EQ_{CFG} is decidable?

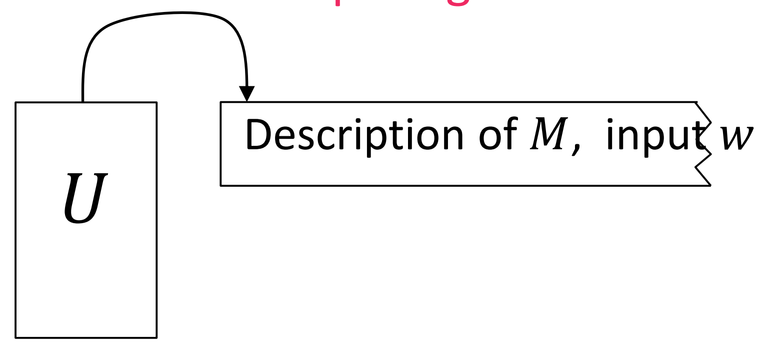
a) Because CFGs are generators and DFAs are recognizers.
b) Because CFLs are closed under union.
c) Because CFLs are not closed under complementation and intersection.

Acceptance problem for Turing machines

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

Theorem: A_{TM} is not decidable

Turing's original
"Universal Computing Machine"



Theorem: A_{TM} is T-recognizable

Proof: The following TM U recognizes A_{TM}

$U =$ "On input $\langle M, w\rangle$

1. Simulate M on input w.
2. *Accept* if M halts and accepts.
3. *Reject* if M halts and rejects.
4. ~~*Reject* if M never halts.~~" Not a legal TM action.

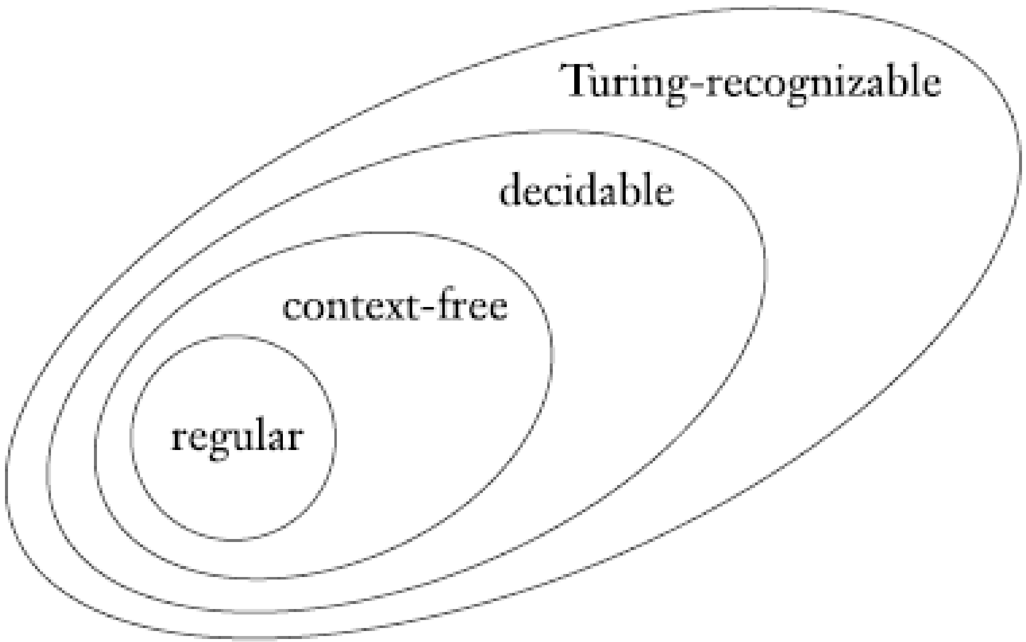
Von Neumann said U inspired the concept of a stored program computer.

Summary

Summary

- Decidable problems concerning regular languages: A_{DFA}, A_{NFA}, E_{DFA}, EQ_{DFA},
- Decidable problems concerning context-free languages: A_{CFG}, E_{CFG}
- Examples of undecidable problems: EQ_{CFG}, $AMBIG_{\text{CFG}}$
- Example of an undecidable but recognizable problem: A_{TM}

Summary



Readings

- Decidable languages: Section 4.1 (Sipser 2013)