Intro to Theory of Computation

CS
332

LECTURE 13

Last time:

- Turing Machine Variants
- Church-Turing Thesis
- Universal Turing Machine
- Decidable languages

Today

- Decidable languages
- Designing deciders

Sofya Raskhodnikova

Recall

- We can encode DFAs, NFAs, regular expressions, PDAs, CFGs, etc into strings of 0s and 1s.
- We defined the following languages:

$A_{DFA} = \{ \langle D,w \rangle \mid D$ is a DFA that accepts string $w \}$

$A_{NFA} = \{ \langle N,w \rangle \mid N$ is an NFA that accepts string $w \}$

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

Recall: Examples of decidable languages

$A_{DFA} = \{ \langle D,w \rangle \mid D \text{ is a DFA that accepts string } w \}$

$A_{NFA} = \{ \langle N,w \rangle \mid N \text{ is an NFA that accepts string } w \}$

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

Decidable languages: more examples

$E_{DFA} = \{ \langle D \rangle \mid D$ is a DFA that recognizes the empty language $\}$

$EQ_{DFA} = \{ \langle D_1, D_2 \rangle \mid D_1, D_2$ are DFAs and $L(D_1) = L(D_2) \}$

$E_{CFG} = \{ \langle G \rangle \mid G$ is a CFG that generates the empty language$\}$

Theorem. E_{DFA} is decidable.

$E_{DFA} = \{ \langle D \rangle \mid D$ **is a DFA that recognizes** $\emptyset$**.**$\}$

Proof: The following TM M decides E_{DFA}**.**

M = `` On input $\langle D \rangle$, where D is a DFA:

1. **Use BFS to determine if an accepting state of D is reachable from from its start state.**
2. **Accept if not. O.w. reject.''**

Theorem. EQ_{DFA} is decidable.

EQ_{DFA} = { $\langle D_1, D_2 \rangle$| D_1, D_2 are DFAs and $L(D_1) = L(D_2)$}

Proof: The following TM M decides EQ_{DFA}.

M = `` On input $\langle D_1, D_2 \rangle$, where D_1, D_2 are DFAs:

1. **Construct a DFA D that recognizes the set difference of $L(D_1)$ and $L(D_2)$.** **(on the board)**
2. **Run the decider for E_{DFA} on <D>.**
3. **If it accepts, accept. O.w. reject."**

Theorem. E_{CFG} is decidable.

E_{CFG} = { $\langle G\rangle$ | G is a CFG that generates no strings}

Proof: **The following TM M decides E_{CFG}.**

M = `` On input $\langle G\rangle$, where G is a CFG:

1. **Mark all terminals in G.**
2. **Repeat until no new variable is marked:**
3. **Mark any variable A where G has a rule A$\to \cdots$ and each variable/terminal on the RHS is already marked.**
4. **Accept if the start variable is unmarked. O.w. reject.''**

Exercises

- Prove that the following language is decidable:

R_{DFA} = { ⟨D,w⟩ | **D is a DFA that rejects string w }**

- Formulate the following problem as a language and prove that it is decidable:

Given a PDA and a string, determine if the PDA accepts the string.

A_{PDA} = { ⟨P,w⟩ | **P is a PDA that accepts string w }**

Can a TM just simulate P on w, accept if it accepts and reject o.w.?

Exercise

A decider for A_{PDA} can, on input <P, w>

A. simulate P on w, accept if it accepts and reject o.w.

B. convert P to an equivalent CFG G and then run a decider for A_{CFG}, accept if it accepts and reject o.w.

C. convert P to an equivalent CFG G and then run a decider for A_{CFG}, accept if it rejects and reject o.w.

D. None of the above.

E. More than one choice above works.

Examples of decidable languages so far

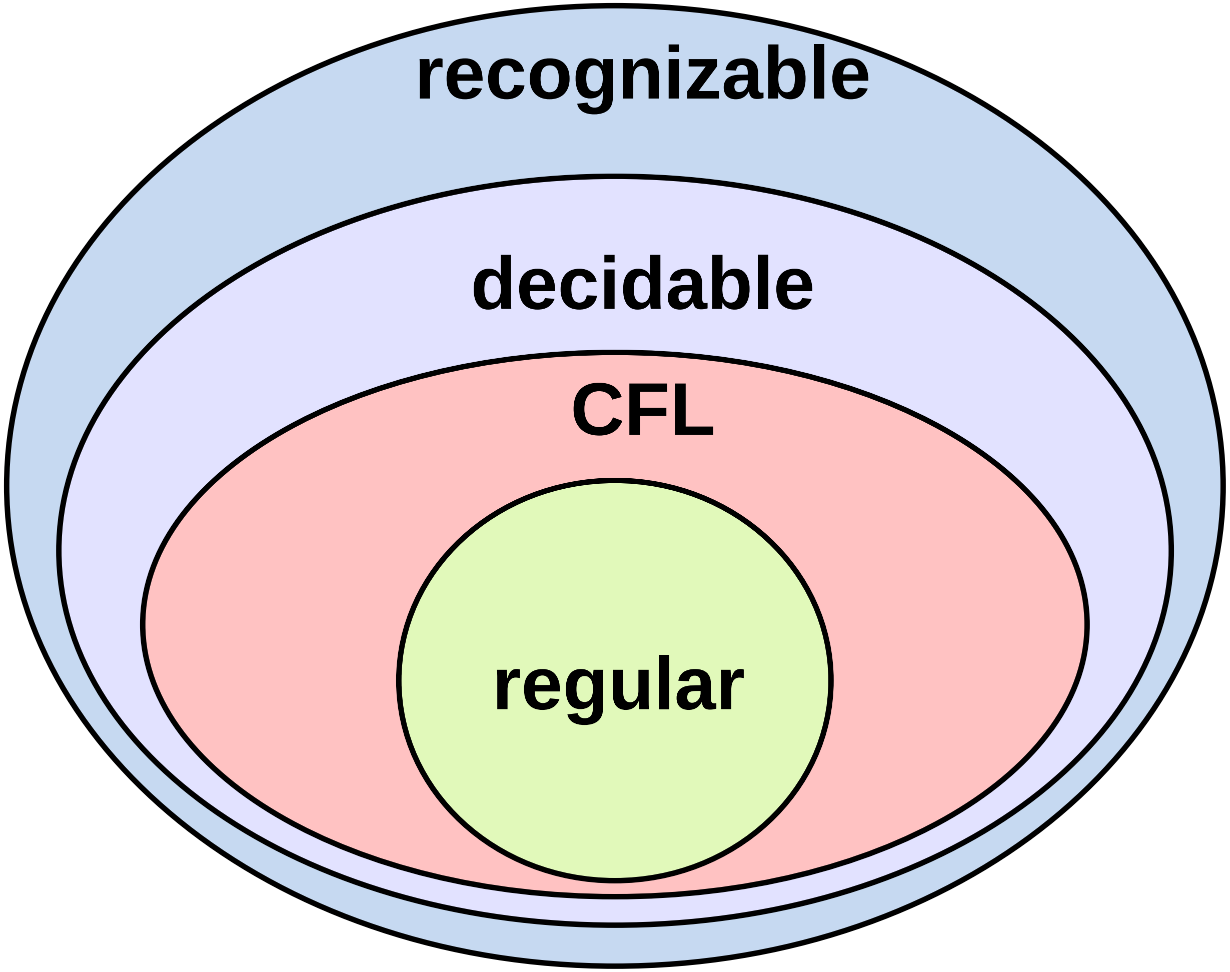
A_{DFA} $= \{\langle D, w\rangle \mid$ D is a DFA that accepts string w $\}$

E_{DFA} $= \{ \langle D\rangle \mid$ D is a DFA and L(D)=∅$\}$

$EQ_{DFA} = \{\langle D_1, D_2\rangle \mid D_1, D_2$ DFAs and $L(D_1) = L(D_2)\}$

A_{DFA}, E_{DFA}, EQ_{DFA}, A_{CFG}, E_{CFG} are decidable.

Classes of languages



Theorem. Every CFL is decidable.

Proof: **Let G be a CFG for L.**

Design a TM M_G that decides L.

- **Is it a good idea to convert G to an equivalent PDA P and have M_G simulate P?**

Exercise

G is a CFG for L. Design a TM M_G that decides L.

Is it a good idea to convert G to an equivalent PDA P and have M_G simulate P?

A. Yes. Why not?

B. No, we can't always convert G to an equivalent PDA.

C. No, P might loop on some inputs.

D. No, because we don't have any input to run P on.

E. None of the above.

Exercise

G is a CFG for L. Design a TM M_G that decides L.

A decider for which language is useful as a subroutine?

A. for $\mathbf{A_{DFA}}$
B. for $\mathbf{E_{DFA}}$
C. for $\mathbf{EQ_{DFA}}$
D. for $\mathbf{A_{CFG}}$
E. for $\mathbf{E_{CFG}}$

CS 332

Theorem. Every CFL is decidable.

Proof: Let G be a CFG for L.

Design a TM M_G that decides L.

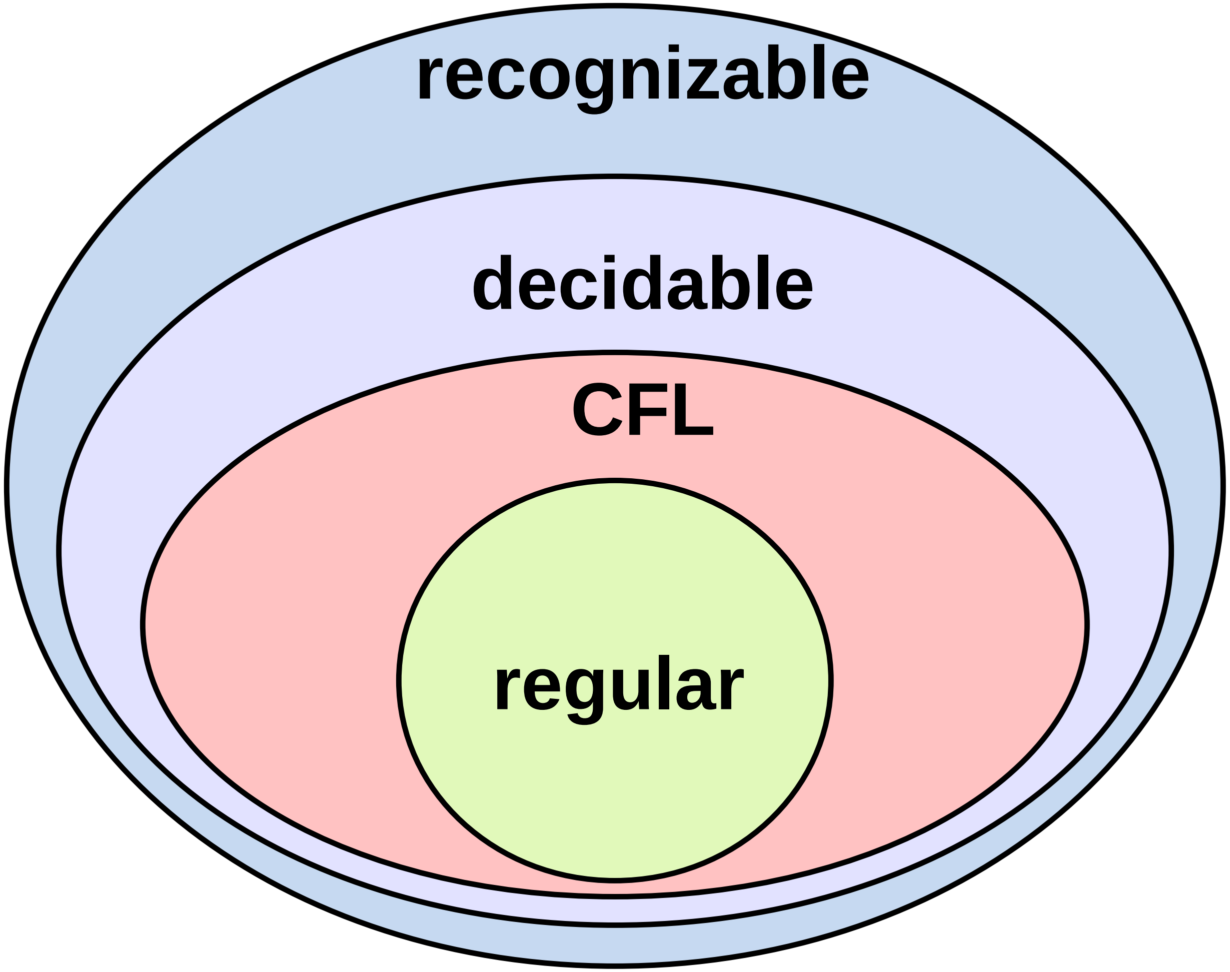
- **Is it a good idea to convert G to an equivalent PDA P and have M_G simulate P?**

M = `` On input w:

1. **Run the decider for A_{CFG} on input <G,w>.**
2. **Accept if it accepts. O.w. reject.”**

CS 332

Classes of languages



Theorem. $\text{INFINITE}_{\text{DFA}}$ is decidable.

$\text{INFINITE}_{\text{DFA}}$ = { ⟨D⟩ | D is a DFA and L(D) is infinite}

Idea: **Let n be the number of states in D.**

L(D) is infinite iff D accepts a string of length $\geq n$.

Proof: **The following TM M decides** $\text{INFINITE}_{\text{DFA}}$.

M = `` On input $\langle D \rangle$, where D is a DFA:

1. **Let n be the number of states in D.**
2. **Let C be a DFA for {w | |w| ≥ n}.**
3. **Build a DFA B for L(C) ∩ L(D).**
4. **Run a decider for E_{DFA} on $\langle B \rangle$.**
5. **Accept if it rejects. O.w. reject."**

Theorem. PAL_{DFA} is decidable.

- Formulate the following problem as a language and prove that it is decidable:

Given a DFA, determine if it accepts some palindrome.

Problems in language theory

A_{DFA} decidable	A_{CFG} decidable	A_{TM} ?
E_{DFA} decidable	E_{CFG} decidable	E_{TM} ?
EQ_{DFA} decidable	EQ_{CFG} ?	EQ_{TM} ?