Computing Theory

COMP 147 (4 units)

Chapter 4: Decidability

Section 4.1: Decidable languages

Decidability Overview

- Every question about regular languages is decidable
- Few questions about context free languages are decidable (most are not)
- Most questions about Turing Machines are not decidable (Some are not Turing recognizable)
- The “Halting Problem” is undecidable

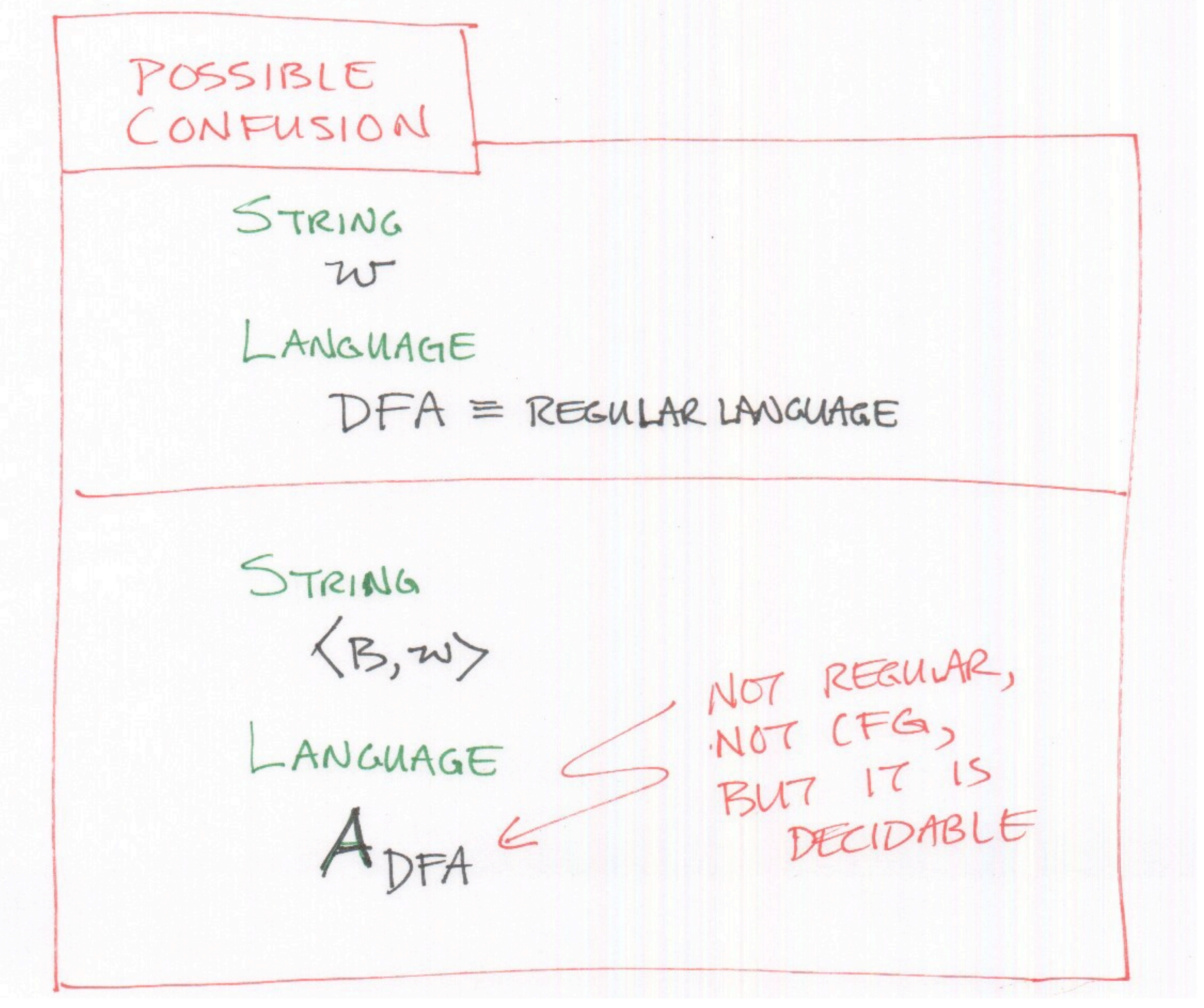
Decidable Problems for Regular Languages: DFAs

$$A_{\text{DFA}} = \{< B, w >| \; B \text{ is a DFA that accepts a given string } w\}$$

- **Acceptance problem for DFAs**
  - Language includes encodings of all DFAs and strings they accept.
  - Showing language is decidable is same as showing the computational problem is decidable.

Decidable Problems for Regular Languages: DFAs

$$A_{DFA} = \{< B, w > \mid B \text{ is a DFA that accepts a given string } w\}$$



Decidable Problems for Regular Languages: DFAs

- **Acceptance problem for DFAs**

$$A_{\text{DFA}} = \{<B, w>|\ B \text{ is a DFA that accepts a given string } w\}$$

  - Language includes encodings of all DFAs and strings they accept.
  - Showing language is decidable is same as showing the computational problem is decidable.
- **Theorem 4.1**: A_{DFA} is a decidable language.
  - **Proof Idea**: Specify a TM M that decides A_{DFA}.
    - M = "On input $<B,w>$, where B is a DFA and w is a string:
      1. If input is valid. Simulate B on input w.
      2. If simulation ends in accept state, *accept*. If it ends in nonaccepting state, *reject*."

Implementation details??

Decidable Problems for Regular Languages: NFAs

- **Acceptance problem for NFAs**

  $$A_{NFA} = \{<B, w> \mid B \text{ is an NFA that accepts a given string } w\}$$

- **Theorem 4.2**: A_{NFA} is a decidable language.
  - **Proof Idea**: Specify a TM N that decides A_{NFA}.
    - N = "On input $<B,w>$, where B is an NFA and w is a string:
      1. Convert NFA B to equivalent DFA C
      2. Run TM M from Theorem 4.1 on input $<C,w>$.
      3. If M accepts, ***accept***. Otherwise, ***reject***."

N uses M as a "subroutine."

Alternatively, could we have modified proof of Theorem 4.1 to accommodate NFAs?

Decidable Problems for Regular Languages: Regular Expressions

- **Acceptance problem for Regular Expressions**

$$A_{\text{REX}} = \{< R, w >| \ R \text{ is a regular expression that generates string } w\}$$

- **Theorem 4.3**: A_{REX} is a decidable language.
  - **Proof Idea**: Specify a TM P that decides A_{REX}.
    - P = "On input <R,w>, where R is a regular expression and w is a string:
      1. Convert regular expression R to equivalent NFA A using Theorem 1.54.
      2. Run TM N from Theorem 4.2 on input <A,w>.
      3. If N accepts, ***accept***. If N rejects, ***reject***."

Overview of Section 4.1

- *Decidable Languages* : to foster later appreciation of undecidable languages
  - Regular Languages
    - Acceptance problem for DFAs
    - Acceptance problem for NFAs
    - Acceptance problem for Regular Expressions
    - **Emptiness testing for DFAs**
    - 2 DFAs recognizing the same language

Decidable Problems for Regular Languages: DFAs

- **Emptiness problem for DFAs**

$$E_{DFA} = \{< A >| A \text{ is a DFA and } L(A) = \emptyset\}$$

- **Theorem 4.4**: E_{DFA} is a decidable language.
  - **Proof Idea**: Specify a TM T that decides E_{DFA}.
    - T = "On input <A>, where A is a DFA:
      1. Mark start state of A.
      2. Repeat until no new states are marked:
      3. Mark any state that has a transition coming into it from any state that is already marked.
      4. If no accept state is marked, *accept*; otherwise, *reject*."

Overview of Section 4.1

- *Decidable Languages*: to foster later appreciation of undecidable languages
  - Regular Languages
    - Acceptance problem for DFAs
    - Acceptance problem for NFAs
    - Acceptance problem for Regular Expressions
    - Emptiness testing for DFAs
    - **2 DFAs recognizing the same language**

Decidable Problems for Regular Languages: DFAs

- **2 DFAs recognizing the same language**

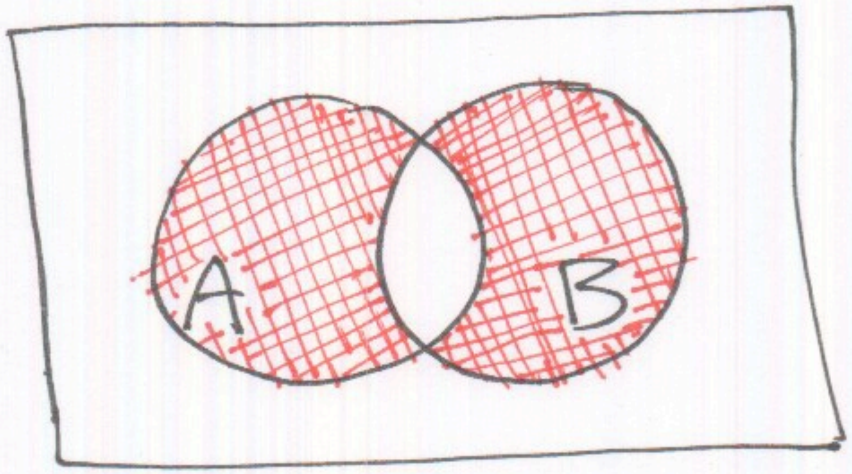
$$\mathrm{EQ}_{\mathrm{DFA}} = \{< A, B >| \ A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$$

- **Theorem 4.5**: $\mathrm{EQ}_{\mathrm{DFA}}$ is a decidable language.

Source: Sipser Textbook

PROOF

Let C ~~is~~ be the "SYMMETRIC DIFFERENCE" between A and B.



"ANYTHING IN A OR B BUT NOT BOTH."

$$C = (A \cap \overline{B}) \cup (\overline{A} \cap B)$$

$\boxed{\text{NOTE}}$ IF $A = B$ THEN THE SYMMETRIC DIFFERENCE WILL BE $\emptyset$.

GIVEN...

A = DFA TO ACCEPT $L(A)$

B = DFA TO ACCEPT $L(B)$

...WE KNOW HOW TO COMBINE DFA's.

$$\overline{L(A)}$$

$$L(A) \cup L(B)$$

$$L(A) \cap L(B)$$

BUILD DFA C TO ACCEPT THE SYMMETRIC DIFFERENCE.

USE THE TM FROM PREVIOUS THEOREM [E_{DFA} ≡ empty language] TO TEST.

F = "On input $\langle A, B\rangle$, where A and B are DFAs:

1. Construct DFA C as described.
2. Run TM T from Theorem 4.4 on input $\langle C\rangle$.
3. If T accepts, *accept*. If T rejects, *reject*."

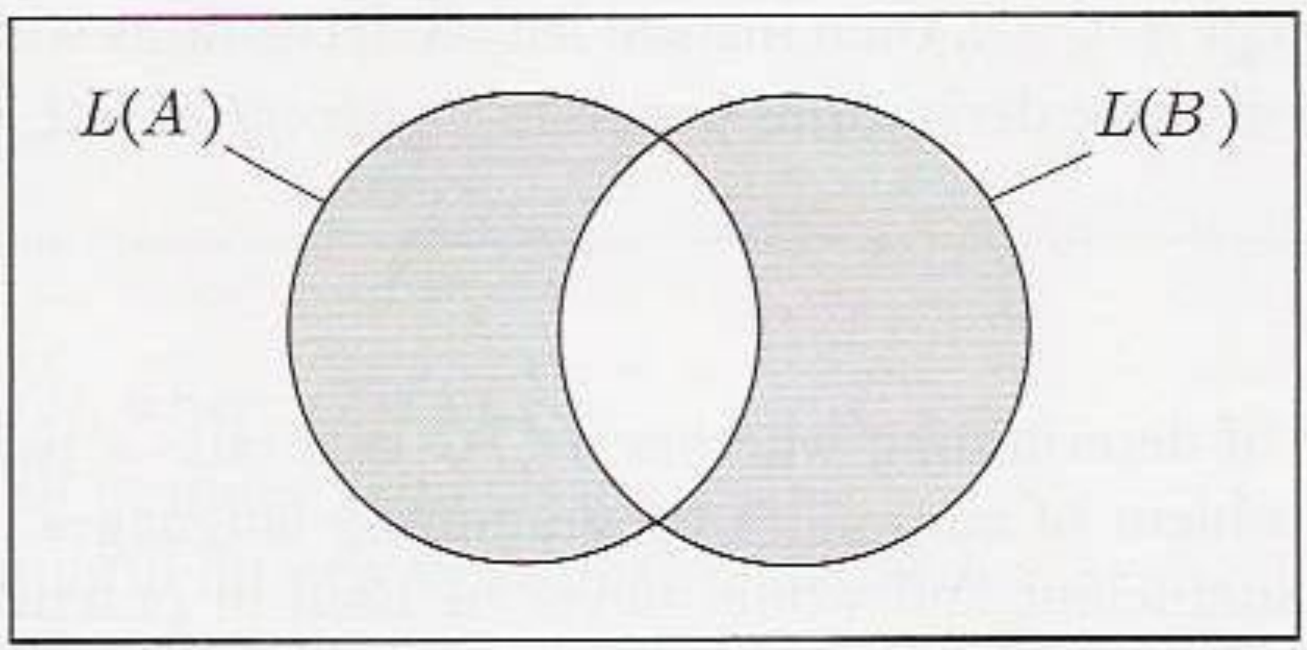


FIGURE 4.6

The symmetric difference of $L(A)$ and $L(B)$

Context-Free Languages

- Given a CFG will it generate a given string w
    - Decidable!
- Given a CFG with generate the empty language (no strings)
    - Decidable!
- Given 2 CFGs do they generate the same language
    - undecidable!
- Is a CFG ambiguous?
    - undecidable!

Decidable Problems for Context-Free Languages: CFGs

- **Does a given CFG generate a given string?**

$$A_{CFG} = \{< G, w >| \, G \text{ is a CFG that generates string } w\}$$

- **Idea#1**
  - Enumerate all leftmost derivations. For each test if generates w
  - Problem: May not halt if w is not in the language

- **Better Idea: Convert grammar to CNF**

  1. Only need to list all derivations with 2n-1 steps (**why?**), where n = length of w.

DERIVATIONS USING CNF GRAMMARS

At each step, the length grows by exactly 1.

$$S \to SS$$
$$S \to a$$

N-1 steps

$$S \underset{1}{\Rightarrow} SS \underset{2}{\Rightarrow} SSS \underset{3}{\Rightarrow} SSSS \underset{4}{\Rightarrow} \underbrace{SSSSS}_{5}$$

... Plus 1 additional step for each terminal symbol.

$$\underset{1}{\Rightarrow} aSSSS \underset{2}{\Rightarrow} aaSSS \underset{3}{\Rightarrow} aaaSS \underset{4}{\Rightarrow} aaaaS$$
$$\underset{5}{\Rightarrow} aaaaa$$

N steps

∴ EVERY DERIVATION HAS EXACTLY

$2N-1$

steps.

Step 2:

Let N be the length of w.

List all derivations of length 2N-1.

(There are only finitely many.)

CHECK EACH DERIVATION TO SEE IF IT GENERATES w.

IF ANY DERIVATION GENERATES w, THEN ACCEPT.
ELSE REJECT.

Decidable Problems for Context-Free Languages: CFGs

- **Does a given CFG generate a given string?**

$$A_{\text{CFG}} = \{<G, w>|\ G \text{ is a CFG that generates string } w\}$$

- **Theorem 4.7**: A_{CFG} is a decidable language.
  - Why is this unproductive: use G to go through all derivations to determine if any yields w?
  - Better Idea…**Proof Idea**: Specify a TM S that decides A_{CFG}.
    - S = "On input <G,w>, where G is a CFG and w is a string:
      1. Convert G to equivalent Chomsky normal form grammar.
      2. List all derivations with $2n$-1 steps, where n = length of w. (Except if n=0, only list derivations with 1 step.)
      3. If any of these derivations yield w, ***accept***; otherwise, ***reject***."

Decidable Problems for Context-Free Languages: CFGs

- **Is the language of a given CFG empty?**

  $$\mathrm{E}_{\mathrm{CFG}} = \{< G >|\ G \text{ is a CFG and } L(G) = \emptyset\}$$

- **Theorem 4.8**: $\mathrm{E}_{\mathrm{CFG}}$ is a decidable language.
  - **Proof Idea**: Specify a TM R that decides $\mathrm{E}_{\mathrm{CFG}}$.
    - R = "On input <G>, 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 rule A->U_1, U_2 … U_k and each symbol U_1, U_2 … U_k has already been marked.
      4. If start variable is not marked, ***accept***; otherwise, ***reject***."

Decidable Problems

Acceptance Tests:

$$A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}.$$

$$A_{\mathsf{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts input string } w\}.$$

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

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

Emptiness Tests:

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

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

Equivalence Tests:

$$EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}.$$