

Theory of Computation

---

**Lecture 06**

***Decidability  
and  
Undecidability***

C SC 473 Automata, Grammars & Languages

---

---

---

---

---

---

---

---

Decidable Problems for Regular Languages

- Theorem 4.1: (Membership/Acceptance Prob. for DFAs)  
 $A_{DFA} = \{ \langle A, w \rangle \mid A \text{ is a DFA and } w \in L(A) \}$  is a decidable language.  
*Pf:* A decider for the language is:
  - $M =$  "On input  $\langle A, w \rangle$ :
    - Simulate  $A$  on  $w$ .
    - If the simulation reaches an accept state, halt and *accept*. If it ends (jams) in a non-accept state, halt and *reject*."

C SC 473 Automata, Grammars & Languages 2

---

---

---

---

---

---

---

---

Decidable—Regular Lang.s (cont'd)

- Theorem 4.2: (Membership/Acceptance Prob. for NFAs)  
 $A_{NFA} = \{ \langle N, w \rangle \mid N \text{ is an NFA and } w \in L(N) \}$  is a decidable language.  
*Pf:* A decider for the language is:
  - $M =$  "On input  $\langle N, w \rangle$ :
    - Use the Rabin-Scott algorithm to convert  $N$  to DFA  $A$
    - Run the Theorem 4.1 algorithm with input  $\langle A, w \rangle$ ."
    - If that algorithm accepts, then accept; otherwise reject."

C SC 473 Automata, Grammars & Languages 3

---

---

---

---

---

---

---

---

Decidable—Regular Lang.s (cont'd)

- Theorem 4.3: (Membership Prob. for RegEx's)  
 $A_{\text{REG}} = \{ \langle R, w \rangle \mid R \text{ is a regular expression and } w \in L(R) \}$  is a decidable language.  
*Pf:* Use the algorithm to convert  $R$  to an equivalent NFA  $N$  and use the algorithm of Theorem 4.2 with input  $\langle N, w \rangle$

C SC 473 Automata, Grammars & Languages

4

---

---

---

---

---

---

---

---

---

---

Decidable—Regular Lang.s (cont'd)

- Theorem 4.4: (*Emptiness Problem for DFAs*)  
 $E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \}$  is a decidable language.  
*Pf:* A decider for the language is:
  - $T =$  "On input  $\langle A \rangle$ :
    - $S \leftarrow \{q_0\}$
    - repeat {  $M \leftarrow S$   
 $S \leftarrow M \cup \{q \mid (\exists a)(\exists p \in M) \delta(p, a) = q\}$
    - } until (  $S = M$  )
    - if  $F \cap M = \emptyset$  accept; otherwise reject."

C SC 473 Automata, Grammars & Languages

5

---

---

---

---

---

---

---

---

---

---

Decidable—Regular Lang.s (cont'd)

- Theorem 4.5: (*Equivalence Problem for DFAs*)  
 $EQ_{\text{DFA}} = \{ \langle A, B \rangle \mid A, B \text{ are DFA and } L(A) = L(B) \}$  is a decidable language.  
*Pf:* Observe that  
 $L(A) = L(B) \iff L(A) \oplus L(B) = \emptyset$  where  
 $L(A) \oplus L(B) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$   
 Because of closure properties, there is an algorithm to construct a DFA  $C$  from  $A, B$  that accepts  $L(A) \oplus L(B)$   
 Use Theorem 4.4 with  $\langle C \rangle$  to test whether  $L(C) = \emptyset$ .  
 If that algorithm accepts, then accept; otherwise, reject.

C SC 473 Automata, Grammars & Languages

6

---

---

---

---

---

---

---

---

---

---

### Decidable Problems for CFLs

- Theorem 4.7: (Membership/Acceptance Prob. for CFGs)  
 $A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG and } w \in L(G) \}$  is a decidable language.  
*Pf:* Chomsky Normal Form parse trees look like:

Pure Binary Tree w/  $n$  leaves has  $n-1$  internal nodes. Add  $n$  terminating rules for a parse tree.

$n$  terminals

C SC 473 Automata, Grammars & Languages 7

---

---

---

---

---

---

---

---

---

---

### Decidable Problems—CFLs (cont'd)

- $S =$  "On input  $\langle G, w \rangle$ :
  - Convert  $G$  to CNF
  - If  $|w| > 0$  try all derivations with  $2|w|-1$  steps. If  $|w|=0$  try the 1-step derivation  $S \Rightarrow_G^1 \epsilon$ .
  - If any derivation generates  $w$ , accept; else reject."
- Corollary (Text Theorem 4.9). Every CFL is a decidable language.  
*Pf:* Let  $A$  be a CFL. We want a decider for it. Let  $G$  be a CFG generating  $A$ . On input  $w$ , run the TM  $S$  above on  $\langle G, w \rangle$  to accept or reject  $w$ .

C SC 473 Automata, Grammars & Languages 8

---

---

---

---

---

---

---

---

---

---

### Decidable—CFLs (cont'd)

- Theorem 4.8: (Emptiness Problem for CFGs)  
 $E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$  is a decidable language.  
*Pf:* A variable  $A$  in a CFG is *productive* (or *co-reachable*) iff  $(\exists w \in \Sigma^+) A \Rightarrow_G^* w$ . So  $L(G) \neq \emptyset$  iff the start variable  $S$  is productive. See Homework 4, Problem 2 for an algorithm to decide whether a variable is productive.
- All above problems also decidable for PDAs: just convert to CFGs.
- What about the Equivalence Problem for CFGs?  
 $EQ_{CFG} = \{ \langle G, H \rangle \mid G, H \text{ are CFGs and } L(G) = L(H) \}$   
 We will show (later) that this problem is UNdecidable.

C SC 473 Automata, Grammars & Languages 9

---

---

---

---

---

---

---

---

---

---

### The Halting Problem

- Although the following language is TM-recognizable, we will show it is not decidable.  
 $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \in L(M) \}$
- This is called the Membership Problem for TMs, and by some authors the Halting Problem for TMs: given a TM  $M$  and string  $w$ , does  $M$  accept  $w$ ?
- Why called "Halting Problem"? Given a TM  $M$ , can always alter it to an equivalent TM  $M'$  such that:  
 $M'$  halts on  $w$  iff  $M'$  accepts  $w$  (iff  $M$  accepts  $w$ ).  
*Pf:* For each undefined transition  $\delta(q, a)$  in  $M$ ,  $M'$  will transition to a state  $q_{loop}$  and loop forever; also  $q_{reject}$  goes to the same loop state
- Thus acceptance can be made synonymous with halting

C SC 473 Automata, Grammars & Languages 10

---

---

---

---

---

---

---

---

---

---

### The Halting Problem (cont'd)

- Thm:  $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \in L(M) \}$  is Turing-recognizable.
- Pf:* Let  $U$  be a UTM. A recognizer for  $A_{TM}$  is:

$\langle M, w \rangle \in L(R) \Leftrightarrow \langle M, w \rangle \in L(U) \Leftrightarrow w \in L(M) \Leftrightarrow \langle M, w \rangle \in A_{TM}$

□

C SC 473 Automata, Grammars & Languages 11

---

---

---

---

---

---

---

---

---

---

### The Halting Problem (cont'd)

- Thm:  $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \in L(M) \}$  is undecidable.
- Pf:* Proof by contradiction. Assume that  $A_{TM}$  is decidable. Then it has a decider; call it  $H$ .  $H$  behaves as follows:

Construct a TM  $D$  that calls  $H$  as a subroutine. On input  $\langle M \rangle$ ,  $D$  runs  $H$  on  $\langle M, \langle M \rangle \rangle$ . That is,  $D$  determines if  $M$  accepts or rejects its own description as input. If  $M$  accepts  $\langle M \rangle$ , then  $D$  rejects; if  $M$  rejects  $\langle M \rangle$ , then  $D$  accepts. Here is the picture of how  $D$  behaves:

C SC 473 Automata, Grammars & Languages 12

---

---

---

---

---

---

---

---

---

---

### The Halting Problem (cont'd)

$\langle M \rangle \in L(D) \Leftrightarrow \langle M \rangle \notin L(M)$

- What happens if we run  $D$  on its own description  $\langle D \rangle$ ? Set  $\langle M \rangle = \langle D \rangle$  in the above. Then  $\langle D \rangle \in L(D) \Leftrightarrow \langle D \rangle \notin L(D)$ . This contradiction shows that decider  $H$  cannot exist.

C SC 473 Automata, Grammars & Languages 13

---

---

---

---

---

---

---

---

---

---

### “Diagonalization”—Why called?

$D$  computes the opposite of the diagonal entries  
 $D(\langle M_i \rangle) = \text{accept} \Leftrightarrow M_i(\langle M_i \rangle) = \text{reject}$

	$M_1$	$M_2$	$M_3$	...	$D$
$\langle M_1 \rangle$	$M_1(\langle M_1 \rangle)$	$M_2(\langle M_1 \rangle)$	$M_3(\langle M_1 \rangle)$	...	
$\langle M_2 \rangle$	$M_1(\langle M_2 \rangle)$	$M_2(\langle M_2 \rangle)$	$M_3(\langle M_2 \rangle)$	...	
$\langle M_3 \rangle$	$M_1(\langle M_3 \rangle)$	$M_2(\langle M_3 \rangle)$	$M_3(\langle M_3 \rangle)$	...	
$\vdots$	$\vdots$	$\vdots$	$\ddots$		
$\langle D \rangle$					$D(\langle D \rangle)$

is  $D(\langle D \rangle) = \text{accept or reject?}$

$D(\langle D \rangle) = \text{accept} \Leftrightarrow D(\langle D \rangle) = \text{reject}$

C SC 473 Automata, Grammars & Languages 14

---

---

---

---

---

---

---

---

---

---

### Decidable vs Recognizable Sets: Basics

- Theorem:  $L$  decidable  $\Rightarrow \bar{L}$  decidable

*Proof:* If  $L$  is decidable, it has a decider  $M$ . The decider halts for every input in either the accepting halt state  $q_{\text{accept}}$  or in the rejecting halt state  $q_{\text{reject}}$ . Construct  $\bar{M}$  from  $M$  as follows: make the accepting halt state the rejecting state and the rejecting halt state the accepting state. Then  $w \in L(\bar{M}) \Leftrightarrow w \notin L(M)$ , that is,  $L(\bar{M}) = \bar{L(M)}$ .

C SC 473 Automata, Grammars & Languages 15

---

---

---

---

---

---

---

---

---

---

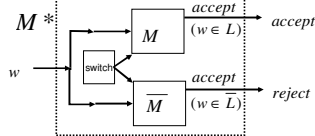
**Decidable vs Recognizable: Basics (Cont.)**

Theorem 4.22:  $L$  is decidable iff both  $L$  &  $\bar{L}$  are Turing-recognizable.

*Proof:* ( $\Rightarrow$ ) Previous theorem.

( $\Leftarrow$ ) Suppose both are recognizable. Let  $M, \bar{M}$  be recognizers for  $L, \bar{L}$ .

Construct  $M^*$  to simulate alternate steps in each recognizer:



Given  $w$  it is eventually accepted by one or the other, so  $M^*$  must halt and either accept or reject; it is a decider.

---

---

---

---

---

---

---

---

---

---

**A Non-TM-Recognizable Set**

- Corollary 4.23:  $\bar{A}_{TM}$  (the complement of  $A_{TM}$ ) is not Turing-recognizable.

*Pf:* By contradiction. We know that  $A_{TM}$  is Turing-recognizable. Suppose  $\bar{A}_{TM}$  is Turing-recognizable. By Theorem 4.22, it follows that  $A_{TM}$  is decidable. Since we know that  $A_{TM}$  is not decidable by Theorem 4.11, this contradiction establishes the result.

Note: What does  $\bar{A}_{TM}$  look like?

$$\bar{A}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \notin L(M) \} \cup J$$

where  $J = \{ \langle (0+1)^*, (0+1)^* \rangle \}$ , which are all the "junk" strings that cannot be of the form  $\langle \text{Coded machine, coded input} \rangle$ . Note that  $J$  is a regular language.

---

---

---

---

---

---

---

---

---

---