

Theory of Computation

Lecture 07

Reduction

C SC 473 Automata, Grammars and Languages

Reduction of One Problem to Another

- Often want to solve a new problem P similar to a problem Q that has already been solved.
- One way of solving P is to transform each instance of P into an instance of the known problem Q , then solve the Q instance, and then use it to obtain a solution to the P instance.
- The solution to P uses the solution to Q as a “subroutine”.
- We often write $P \leq Q$ for “ P is reducible to Q ”
- Ex: *Squaring* \leq *Multiplication*: $a^2 = a \times a$
- Ex: *Multiplication* \leq *Squaring*:

$$a \times b = ((a + b)^2 - a^2 - b^2) / 2$$
- Ex: DFA Equivalence \leq DFA Emptiness

$$L(A) = L(B) \Leftrightarrow L(A \oplus B) = \emptyset$$

C SC 473 Automata, Grammars and Languages 2

Using Reduction to Prove “Difficulty”

- If $P \leq Q$ and P is known to be “hard to solve”, then Q must be hard to solve too.
- For example[†], if $P \leq Q$ and P is undecidable, then Q must also be undecidable. For if Q is decidable, we can use the reduction $P \leq Q$ to construct a decider for P ; contradiction.
- Ex: We will show by reduction that the problem
 $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts input } w\}$
 is reducible to the problem
 $HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$
- The undecidability of A_{TM} will imply the undecidability of $HALT_{TM}$

[†]Here \leq stands for *many-one* or *mapping reduction* denoted \leq_m . It will be defined precisely later.

C SC 473 Automata, Grammars and Languages 3

Undecidability via Reductions: Halting

- HALTING PROBLEM
 $HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \}$
- ACCEPTANCE (MEMBERSHIP) PROBLEM
 $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts input } w \}$
- Thm 5.1: $HALT_{TM}$ is undecidable.
Pf: We show that $A_{TM} \leq HALT_{TM}$, so that if we had a decider for $HALT_{TM}$ we could build a decider for A_{TM} . This contradicts the undecidability of A_{TM} , and so $HALT_{TM}$ must be undecidable.

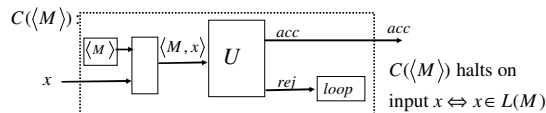
Assume, contrary to what is to be proved, that $HALT_{TM}$ has a decider R . Following is a visual proof that A_{TM} is reducible to $HALT_{TM}$.

C SC 473 Automata, Grammars and Languages

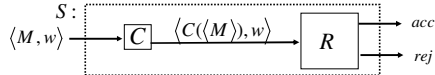
4

Undecidability via Reductions (cont.)

Consider a “compiler” (algorithm) C that given $\langle M \rangle$ constructs a new TM $C(\langle M \rangle)$ as follows:



- Reduction: use this and R to build a decider for A_{TM}



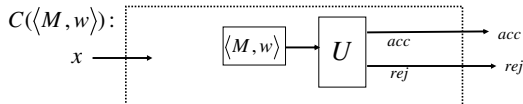
$\langle M, w \rangle \in L(S) \Leftrightarrow \langle C(\langle M \rangle), w \rangle \in L(R) \Leftrightarrow$
 $C(\langle M \rangle)$ halts on input $w \Leftrightarrow w \in L(M) \therefore L(S) = A_{TM}$
 So S is a decider for A_{TM} . Contradiction \Rightarrow theorem.

C SC 473 Automata, Grammars and Languages

5

Undecidability: Empty Tape Acceptance

- Thm: The EMPTY-TAPE-ACCEPTANCE problem is undecidable: $ET_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } \epsilon \in L(M) \}$
Pf: We will show that $A_{TM} \leq ET_{TM}$. Consider a “compiler” C that given $\langle M, w \rangle$ constructs TM $C(\langle M, w \rangle)$:



$C(\langle M, w \rangle)$ accepts the empty tape $\Leftrightarrow w \in L(M)$

In fact: $C(\langle M, w \rangle)$ accepts $\Sigma^* \Leftrightarrow w \in L(M)$
 $C(\langle M, w \rangle)$ accepts $\emptyset \Leftrightarrow w \notin L(M)$

C SC 473 Automata, Grammars and Languages

6

Empty Tape Acceptance (cont'd)

- Reduction: assume a decider R for ET_{TM} . We construct a decider from it for A_{TM} .

E :

$\langle M, w \rangle \in L(E) \Leftrightarrow \langle C(\langle M, w \rangle) \rangle \in L(R)$
 $\Leftrightarrow \varepsilon \in L(C(\langle M, w \rangle)) \Leftrightarrow w \in L(M) \quad \therefore A_{TM} = L(E)$

- E is a decider for A_{TM} . Contradiction.

C SC 473 Automata, Grammars and Languages 7

Undecidability: Empty Set Acceptance

- Thm: The EMPTY-SET-ACCEPTANCE problem is undecidable: $E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}$
- Pf: We will show that $\bar{A}_{TM} \leq E_{TM}$. We reduce the complement of A_{TM} to this problem. Consider a "compiler" C that given $\langle M, w \rangle$ constructs TM $C(\langle M, w \rangle)$:

$L(C(\langle M, w \rangle)) = \begin{cases} \Sigma^* & \text{if } w \in L(M) \\ \emptyset & \text{if } w \notin L(M) \end{cases}$

C SC 473 Automata, Grammars and Languages 8

Empty Set Acceptance (cont'd)

- Reduction: assume a decider R for E_{TM} . We construct a decider from it for \bar{A}_{TM} .

E :

$\langle M, w \rangle \in L(E) \Leftrightarrow \langle C(\langle M, w \rangle) \rangle \in L(R)$
 $\Leftrightarrow L(C(\langle M, w \rangle)) = \emptyset \Leftrightarrow w \notin L(M) \quad \therefore \bar{A}_{TM} = L(E)$

- E is a decider for \bar{A}_{TM} . Contradiction \Rightarrow theorem

C SC 473 Automata, Grammars and Languages 9

Undecidability: Regular Set Acceptance

- Thm 5.3: The REGULAR-SET-ACCEPTANCE problem is undecidable:
 $REGULAR_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular} \}$

Pf: We will show that $A_{TM} \leq REGULAR_{TM}$. Consider a "compiler" C that given $\langle M, w \rangle$ constructs TM $C(\langle M, w \rangle)$:

$$L(C(\langle M, w \rangle)) = \begin{cases} \Sigma^* & \text{if } w \in L(M) \\ \{0^n 1^n \mid n \geq 0\} & \text{if } w \notin L(M) \end{cases}$$

C SC 473 Automata, Grammars and Languages 10

Regular Set Acceptance (cont'd)

- Reduction: assume a decider R for $REGULAR_{TM}$. We construct a decider from it for A_{TM} .

$$L(C(\langle M, w \rangle)) = \begin{cases} \Sigma^* & \text{if } w \in L(M) \\ \{0^n 1^n\} & \text{if } w \notin L(M) \end{cases}$$

$\langle M, w \rangle \in L(E) \Leftrightarrow \langle C(\langle M, w \rangle) \rangle \in L(R)$
 $\Leftrightarrow L(C(\langle M, w \rangle)) = \Sigma^* \Leftrightarrow w \in L(M) \quad \therefore A_{TM} = L(E)$

- E is a decider for A_{TM} . Contradiction \Rightarrow theorem

C SC 473 Automata, Grammars and Languages 11

Mapping Reduction: Motivation

$$L(S) \leq_m L(R)$$

- Halting Problem $L(S) = A_{TM}$ $L(R) = HALT_{TM}$
 $\langle M, w \rangle \in L(S) \Leftrightarrow \langle C(\langle M \rangle), w \rangle \in L(R)$
- Empty-Tape Acceptance Problem $L(E) = A_{TM}$ $L(R) = ET_{TM}$
 $\langle M, w \rangle \in L(E) \Leftrightarrow \langle C(\langle M \rangle), w \rangle \in L(R)$
- Empty-Set Acceptance Problem $L(E) = \bar{A}_{TM}$ $L(R) = E_{TM}$
 $\langle M, w \rangle \in L(E) \Leftrightarrow \langle C(\langle M \rangle), w \rangle \in L(R)$
- Regular-Set Acceptance Problem $L(E) = A_{TM}$ $L(R) = REGULAR_{TM}$
 $\langle M, w \rangle \in L(E) \Leftrightarrow \langle C(\langle M \rangle), w \rangle \in L(R)$

C is an algorithm in each case

C SC 473 Automata, Grammars and Languages 12

TMs can Act as Recognizers or Transducers

- Defn 5.17: A function $f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function**† if \exists a TM transducer M such that on every input w , M halts with $f(w)$ on its tape. Such a TM is called an **algorithm**.

Compare & contrast this definition with:

- Defn 3.6: A language $L \subseteq \Sigma^*$ is a **decidable language** if \exists a TM recognizer M such that on every input w , if $w \in L$ it halts with “accept” and if $w \notin L$ it halts with “reject”. Such a TM is called a **decider**.
 - A recognizer can be viewed as a special case of a transducer that prints only a 1 or 0
 - A language can be viewed as a special case $L : \Sigma^* \rightarrow \{0,1\}$ of a function that returns a boolean value.

†Also called *total computable function*. “Total” means “defined for all arguments w ”.

C SC 473 Automata, Grammars and Languages 13

Function:Set :: Transducer:Recognizer

Functions $f : \Sigma^* \rightarrow \Sigma^*$	Sets (Languages) $L \subseteq \Sigma^*$
Computable † f \exists TM transducer $M \ni$ <ul style="list-style-type: none"> $(\forall w) M$ on w halts & prints $M(w)$ $(\forall w) M(w) = f(w)$ M is called an algorithm	Decidable L \exists TM recognizer $M \ni$ <ul style="list-style-type: none"> $(\forall w) M$ on w halts & prints 0,1 $(\forall w) M(w) = 1 \iff w \in L$ M is called a decider ‡
Partial † Computable f \exists TM transducer $M \ni$ <ul style="list-style-type: none"> $(\forall w) M$ on w halts $\iff f(w)$ defined $(\forall w) M(w) = f(w)$ M is called a procedure	Recognizable L \exists TM recognizer $M \ni$ <ul style="list-style-type: none"> $(\forall w) M$ on w halts $\iff w \in L$ $(\forall w) M(w) = 1 \iff w \in L$ M is called a recognizer

C SC 473 Automata, Grammars and Languages 14

Mapping Reduction: Definition

$L(M_A) = A \leq_m L(M_B) = B$
 $A \leq_m B$

- All have the same pattern $x \in A \iff f(x) \in B$
 - f must be a **computable function**
 - The “compiler” must be an **algorithm**
- Defn 5.20: Language A is **mapping reducible** to language B iff \exists a computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that $(\forall x) x \in A \iff f(x) \in B$. Function f is called the **reduction** of A to B .

C SC 473 Automata, Grammars and Languages 15

Mapping Reduction: Definition (cont'd)

$A = L(M_A)$
 $B = L(M_B)$

$A \leq_m B$

- Picture: $f: \Sigma^* \rightarrow \Sigma^*$

$x \in A \Leftrightarrow f(x) \in B$
 equivalent to
 $x \in \bar{A} \Leftrightarrow f(x) \in \bar{B}$
 $x \in A \Rightarrow f(x) \in B \wedge x \in \bar{A} \Rightarrow f(x) \in \bar{B} \quad \bar{A} \leq_m \bar{B}$

C SC 473 Automata, Grammars and Languages 16

Reduction (cont.)

- Thms 5.22, 5.23, 5.28, 5.29: Let $A \leq_m B$. Then
 - B decidable $\Rightarrow A$ decidable
 - A undecidable $\Rightarrow B$ undecidable
 - B recognizable $\Rightarrow A$ recognizable
 - A non-recognizable $\Rightarrow B$ non-recognizable

- If you want to show a problem P is easier than problem Q then reduce P to Q

- If you want to show a problem P is harder than problem Q then reduce Q to P

C SC 473 Automata, Grammars and Languages 17

Reductions

- Thm: $A \leq_m B \Leftrightarrow \bar{A} \leq_m \bar{B}$
- Ex: $\bar{A}_{TM} \leq_m E_{TM} \Leftrightarrow A_{TM} \leq_m \bar{E}_{TM}$
- Reduction: major method for showing unsolvability or non-recognizability:
 - Goal: to show L_{TEST} is not recognizable [not decidable]
 - Known: L_{BAD} is not recognizable [not decidable]
 - Strategy: reduce L_{BAD} to L_{TEST} ($L_{BAD} \leq_m L_{TEST}$)
 - Method: build *computable* "translator" f to accept L_{BAD} assuming we have a recognizer [decider] for L_{TEST}

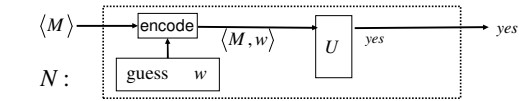
C SC 473 Automata, Grammars and Languages 18

Undecidable via Reductions

- Thm: The NONEMPTY-SET-ACCEPTANCE problem is TM-recognizable but not decidable: $NE_{TM} = \{ \langle M \rangle \mid L(M) \neq \emptyset \}$

Pf: It is easy to see that $\overline{E_{TM}} \leq_m NE_{TM}$ since $w \in \overline{E_{TM}} \Leftrightarrow w \in (0+1)^* \in NE_{TM}$. So NE_{TM} cannot be decidable.

An acceptor for NE_{TM} is the following nondeterministic TM



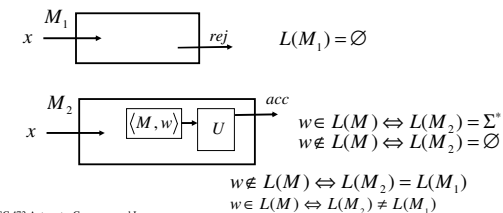
$$\langle M \rangle \in L(N) \Leftrightarrow (\exists w) w \in L(M) \Leftrightarrow L(M) \neq \emptyset \Leftrightarrow \langle M \rangle \in NE_{TM}$$

Non-Recognizable via Reductions

- Thm: The EQUIVALENCE problem is not recognizable: $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs} \wedge L(M_1) = L(M_2) \}$

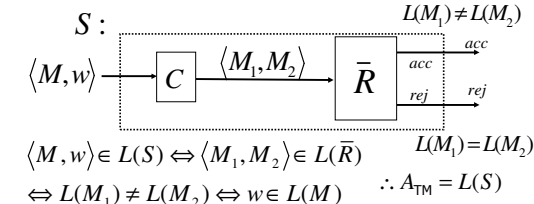
Pf: We show that EQ_{TM} is not recognizable by showing that $A_{TM} \leq_m EQ_{TM}$ (which means that $\overline{A_{TM}} \leq_m EQ_{TM}$).

The reducing function $\langle M, w \rangle \xrightarrow{c} \langle M_1, M_2 \rangle$ generates a pair of TMs with the following behaviors:



Non-recognizable via \leq_m (cont'd)

- Reduction: assume a decider \overline{R} for $\overline{EQ_{TM}}$. We construct a decider from it for A_{TM} .



$\langle M, w \rangle \in L(S) \Leftrightarrow \langle M_1, M_2 \rangle \in L(\overline{R}) \quad L(M_1) = L(M_2)$
 $\Leftrightarrow L(M_1) \neq L(M_2) \Leftrightarrow w \in L(M) \quad \therefore A_{TM} = L(S)$

Non-Recognizable via Reductions (cont'd)

- Exercise: The FINITE-SET-ACCEPTANCE problem is not TM-recognizable: $F_{TM} = \{\langle M \rangle \mid L(M) \text{ is finite}\}$
Proof: Show that $\bar{A}_{TM} \leq_m F_{TM}$
- Exercise: The INFINITE-SET-ACCEPTANCE problem is not TM-recognizable: $I_{TM} = \{\langle M \rangle \mid L(M) \text{ is infinite}\}$
Proof: Show that $\bar{A}_{TM} \leq_m I_{TM}$

C SC 473 Automata, Grammars and Languages 22

Equivalence (\equiv_m) and Completeness

- Definition: Let C be a class of sets. A set A is *mapping reduction complete* (\leq_m -complete) in C iff
 - $A \in C$
 - $\forall B \in C \quad B \leq_m A$
 Remark: "A is complete" says A is a "hardest problem in C"
 A is *mapping equivalent* to B ($A \equiv_m B$) iff $A \leq_m B$ & $B \leq_m A$
- Fact: all complete sets in C are mapping equivalent
- Exercise: $A_{TM} \equiv_m HALT_{TM} \equiv_m NE_{TM}$
 $\bar{A}_{TM} \equiv_m \bar{HALT}_{TM} \equiv_m \bar{E}_{TM}$

C SC 473 Automata, Grammars and Languages 23

Completeness

- Theorem: A_{TM} is complete in the class **TM** of Turing-recognizable sets.
Proof: A_{TM} is accepted by U , so is in **TM**.
 To show completeness, let B be any Turing-recognizable set in **TM**. Then \exists a TM M_B such that $B = L(M_B)$. Define the computable function

$$c(x) = \langle M_B, x \rangle$$
 Then

$$x \in B \Leftrightarrow x \in L(M_B) \Leftrightarrow \langle M_B, x \rangle \in A_{TM} \Leftrightarrow c(x) \in A_{TM}$$

$$\therefore B \leq_m A_{TM}$$
 Since B was chosen arbitrarily in **TM** the result follows
- Exercise: Show the Halting Prob. $HALT_{TM}$ is complete in **TM**

C SC 473 Automata, Grammars and Languages 24
