Lecture 6 Recap: Reductions

Mika Göös

EPFL School of Computer and Communication Sciences

Reductions

Informally

Reducibility Use knowledge about complexity of one language to reason about the complexity of other in an easy way

Reduction Way to show that to solve one problem (A), it is sufficient to solve another problem (B)

A reduces to B...

- Many examples (e.g. calculate area of rectangle reduces to calculate its width and height)
- Reductions quantify relative hardness of problems
  - If problem B is easy then problem A is easy too
  - If problem A is hard then problem B is hard too

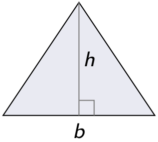
A reduction f from triangle to rectangle area

Triangle-Area $=$

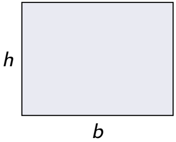
$\{\langle h, b, t\rangle \in \{0,1\}^* \mid h * b/2 \geq t\}$

Rectangle-Area $=$

$\{\langle h, b, r\rangle \in \{0,1\}^* \mid h * b \geq r\}$

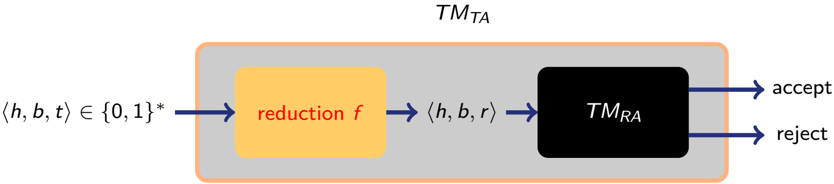


In language if area is at least t

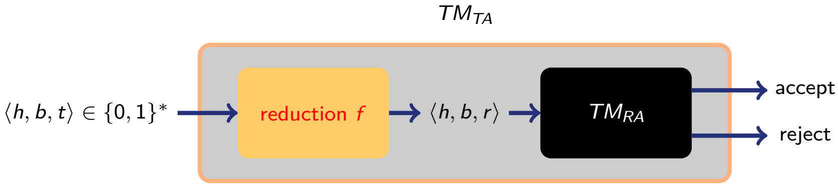


In language if area is at least r

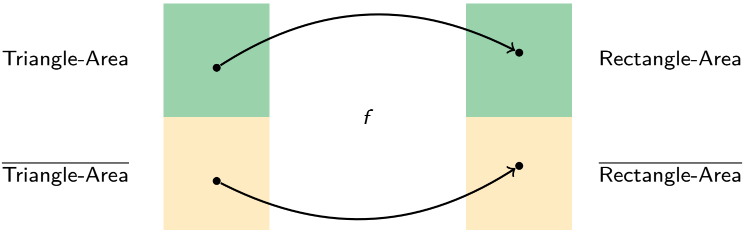
Suppose we have a Turing machine TM_{RA} for deciding Rectangle-Area. How can we use it form a decider TM_{TA} for Triangle-Area?



A reduction f from triangle to rectangle area



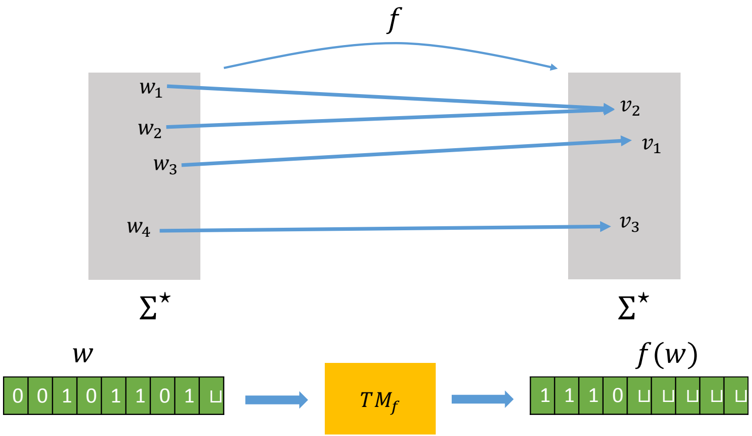
Define $f(\langle h, b, t \rangle) = \langle h, b, 2t \rangle$. Then



- f does not use the knowledge whether input is in language or not
- f can be computed with a TM

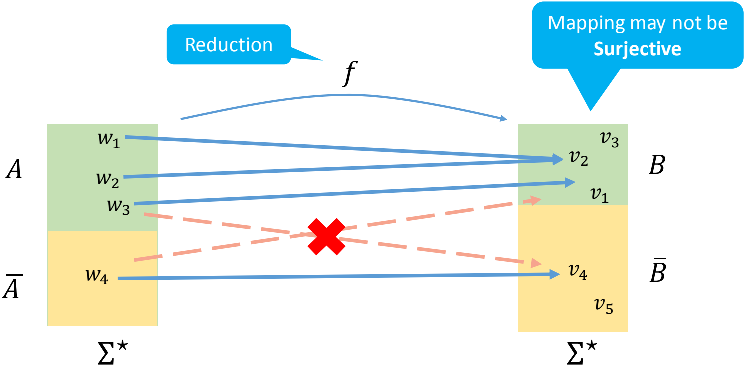
Mapping Reductions

Reductions Part 1: Computability



Definition: A function $f : \Sigma^* \to \Sigma^*$ is a *computable function* if some TM M, on every input w halts with just $f(w)$ on its tape

Reductions Part 2: Correctness



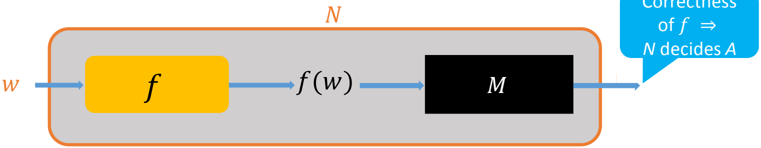
Definition: Language A is **mapping reducible** to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, such that for every $w \in \Sigma^*$:

$$w \in A \Leftrightarrow f(w) \in B$$

Theorem: If $A \leq_m B$ and B is decidable, then A is decidable

Proof:

- Assume that M is a decided for B and f is a reduction from A to B
- Let N be a TM as follows:



- $N =$ "On input w :
  1. Compute $f(w)$
  2. Run M on input $f(w)$ and output whatever M outputs"

Computability of f ⇒ N is a decider

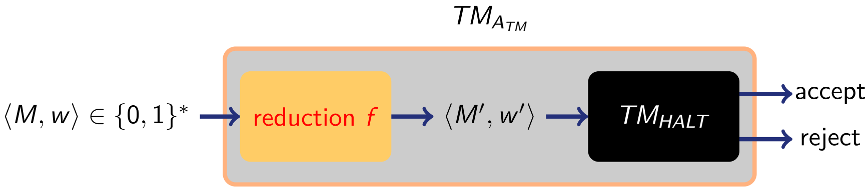
Corollary: If $A \leq_m B$ and A is undecidable, then B is undecidable

Examples of Reductions

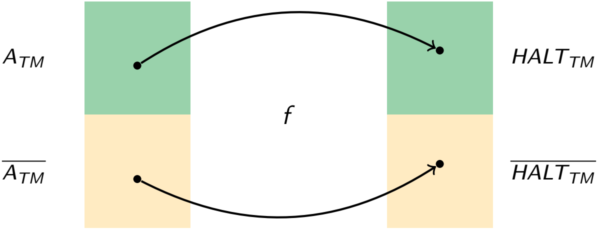
$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$$
$$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$$

Theorem: $A_{TM} \leq_m HALT_{TM}$

Proof idea:



Define computable f such that



Theorem: $A_{TM} \leq_m HALT_{TM}$ ($\Rightarrow HALT_{TM}$ is undecidable)

- Let us define a function f as follows
- Given an input $x = \langle M, w \rangle$, return $f(x) = \langle M', w \rangle$, where
- M' ="On input y :

f has to write down the code of M' only.
It does not run M'!
(M' might loop for some inputs.)

1. Run M on w;
2. **If M rejects w, enter infinite loop.** Otherwise, accept y"

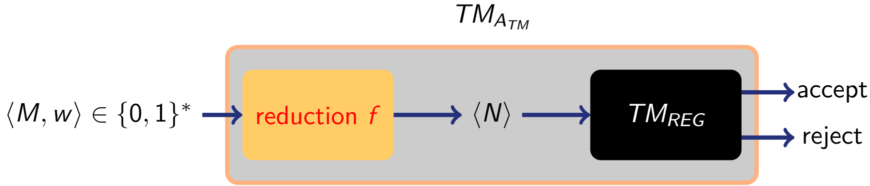
- Check that f is computable

- $\Rightarrow$ If $\langle M, w \rangle \in A_{TM}$ then M' halts on w.
Thus, $\langle M', w \rangle \in HALT_{TM}$
- $\Leftarrow$ If $\langle M, w \rangle \notin A_{TM}$ then either (1) will never halt or M rejects w and (2) will ensure no halting. Thus, $\langle M', w \rangle \notin HALT_{TM}$

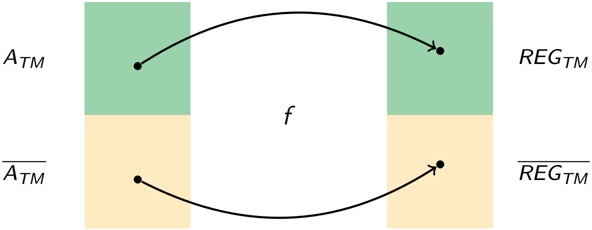
$REG_{TM} = \{\langle N \rangle \mid L(N) \text{ is a regular language}\}$

Theorem: REG_{TM} is undecidable

Proof idea: reduction from A_{TM}



Define computable f such that

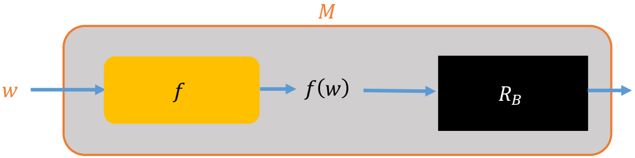


Theorem: $A_{TM} \leq_m REG_{TM}$ ($\Rightarrow REG_{TM}$ is undecidable)

- Let us define a function f as follows
- Given an input $x = \langle M, w \rangle$, return $f(x) = \langle M' \rangle$, where
- M' ="On input y :
  1. if $y \in B = \{0^n 1^n : n \geq 0\}$, then accept y
  2. Run M on w, and accept y iff M accepts w"
- Check that f is computable

- $\Rightarrow$ If $\langle M, w \rangle \in A_{TM}$ then M' accepts all inputs.
  Thus, $L(M') = \{0, 1\}^*$ is regular — $\langle M' \rangle \in REG_{TM}$
- $\Leftarrow$ If $\langle M, w \rangle \notin A_{TM}$ then M does not accept w.
  Thus $L(M') = B$ is non-regular — $\langle M' \rangle \notin REG_{TM}$

Theorem: If $A \leq_m B$ and B is recognizable then A is recognizable



Definition of M · R_B is a recognizer for B · $A \leq_m B$

M accepts w $\Leftrightarrow$ R_B accepts $f(w)$ $\Leftrightarrow$ $f(w) \in B$ $\Leftrightarrow$ $w \in A$

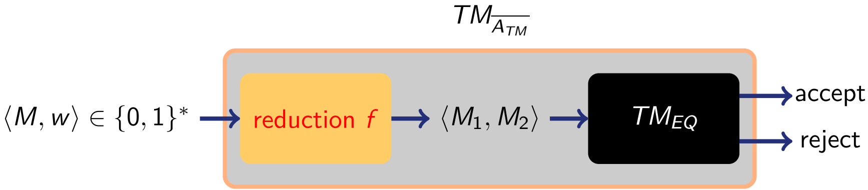
$\Rightarrow$ ***M* is a recognizer for *A***

Corollary: If $A \leq_m B$ and A is unrecognizable then B is unrecognizable

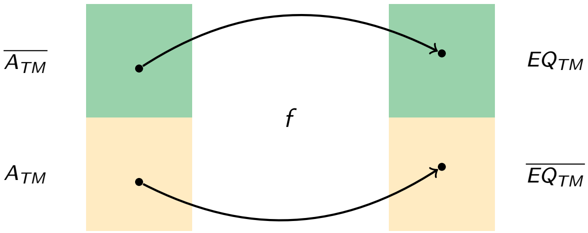
$$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$$

Theorem: EQ_{TM} is unrecognizable

Proof idea: reduction from $\overline{A_{TM}}$



Define computable f such that



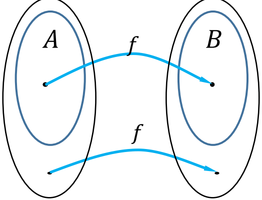
$$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\}$$

Theorem: $\overline{A_{TM}} \leq_m EQ_{TM}$ ($\Rightarrow EQ_{TM}$ is unrecognizable)

- Let us define a function f as follows
- Given an input $x = \langle M, w\rangle$, return $f(x) = \langle M_1, M_2\rangle$, where
- $M_1 =$"On input y :
  1. Run M on w (ignore the input y)
  2. If M accepts then accept, else enter an infinite loop
- $M_2 =$"On input y :
  1. Reject y

- $\Rightarrow$ If $\langle M, w\rangle \in \overline{A_{TM}}$ then M_1 loops on all inputs.
  Thus, $L(M_1) = \emptyset = L(M_2)$ — $\langle M_1, M_2\rangle \in EQ_{TM}$
- $\Leftarrow$ If $\langle M, w\rangle \notin \overline{A_{TM}}$ then M accepts w and hence M_1 accepts every string. Thus $L(M_1) = \Sigma^* \neq \emptyset = L(M_2)$ — $\langle M_1, M_2\rangle \notin EQ_{TM}$

Summary of Reductions



Definition: A function $f : \Sigma^* \to \Sigma^*$ is a *computable function* if some TM M, on every input w halts with just $f(w)$ on its tape

Definition: Language A is **mapping reducible** to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, such that for every $w \in \Sigma^*$:

$$w \in A \Leftrightarrow f(w) \in B$$

f is called a **reduction** of A to B

Theorem: If $A \leq_m B$ and B is decidable (recognizable), then A is decidable (recognizable)

Corollary: If $A \leq_m B$ and A is undecidable (unrecognizable) then B is undecidable (unrecognizable)