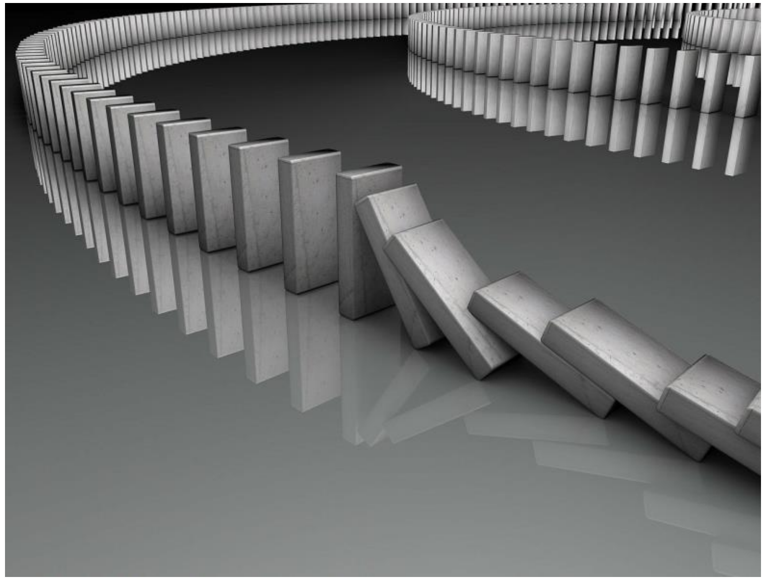
CMSC 28100

Introduction to Complexity Theory

Spring 2024
Instructor: William Hoza

Which languages are decidable?

Undecidability



- First, we proved that SELF-REJECTORS is undecidable
- Then, we used the fact that SELF-REJECTORS is undecidable to prove that HALT is undecidable
- Then, we used the fact that HALT is undecidable to prove that other interesting languages are undecidable

The “acceptance problem”

- Let $\mathrm{A_{TM}} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts input } w\}$
- **Claim:** $\mathrm{A_{TM}}$ is undecidable
- **Proof by contradiction:** Assume that A decides $\mathrm{A_{TM}}$
- Let’s design an algorithm that decides HALT. Given $\langle M, w\rangle$:

1. Construct $\langle M'\rangle$, where M' is a modified version of M in which all rejecting transitions have been changed into accepting transitions
2. Simulate A on $\langle M', w\rangle$. If it accepts, accept; if it rejects, reject.

Reductions

- The proof strategy we have been using is called a reduction
- A reduction is a way of relating one problem to another problem

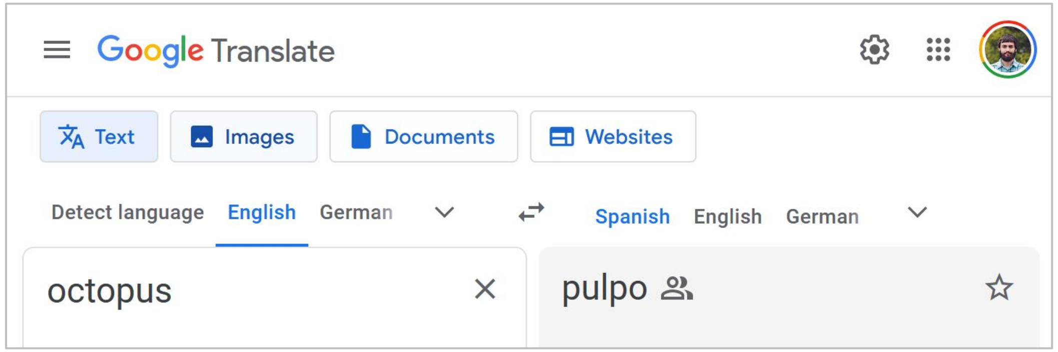
Informal definition:

"Problem A reduces to problem B"

means

"A solution to problem B would imply a solution to problem A"

Reductions



- **Example:** Suppose I bring my daughter to a zoo in Mexico. She asks, "Do they have octopuses? Do they have camels? Do they have gorillas? Do they have …"
- My job is to solve problem A: "Given an animal name in English, determine whether it's at the zoo"
- A helpful zoo employee can solve problem B: "Given an animal name in Spanish, determine whether it's at the zoo"
- We can reduce problem A to problem B by translating from English to Spanish

Mapping reductions

- There are multiple kinds of reductions in computer science
- In this course, we will focus on a kind of reduction called a "mapping reduction"

Mapping reductions

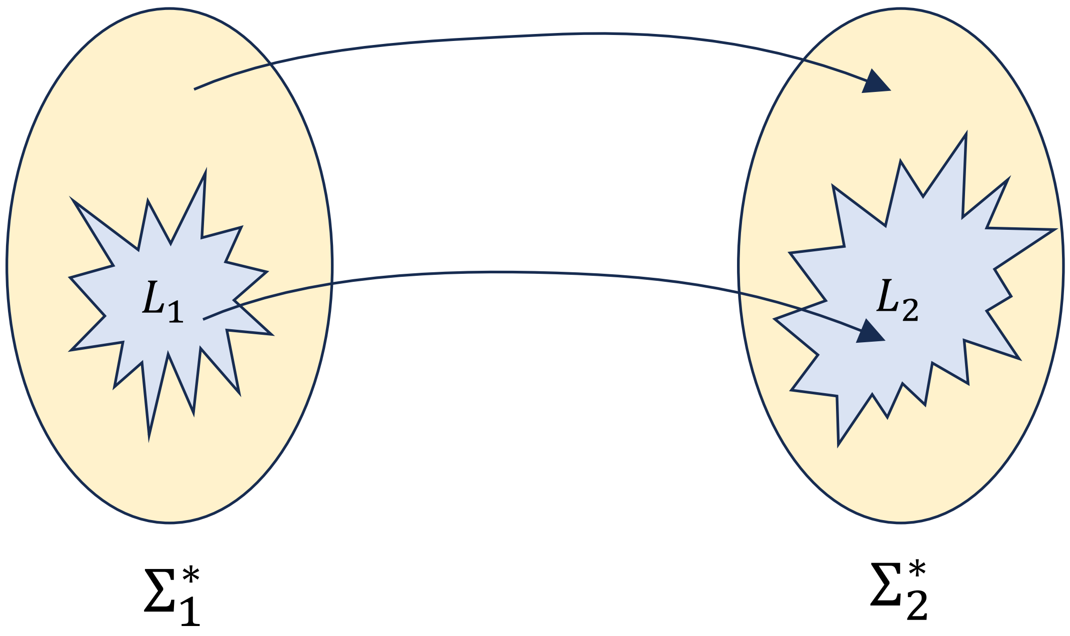
$f(\text{octopus}) = \text{pulpo}$

M = Google Translate

- Let L_1 and L_2 be languages over the alphabets Σ_1 and Σ_2 respectively
- **Definition:** A mapping reduction from L_1 to L_2 is a function $f : \Sigma_1^* \to \Sigma_2^*$ such that
  - For every $w \in L_1$, we have $f(w) \in L_2$ "YES maps to YES"
  - For every $w \in \Sigma_1^* \setminus L_1$, we have $f(w) \notin L_2$ "NO maps to NO"
  - The function f is computable, i.e., there exists a Turing machine M such that for every $w \in \Sigma_1^*$, M halts on input w with $\Diamond f(w)$ written on its tape (followed by blanks)

Mapping reductions

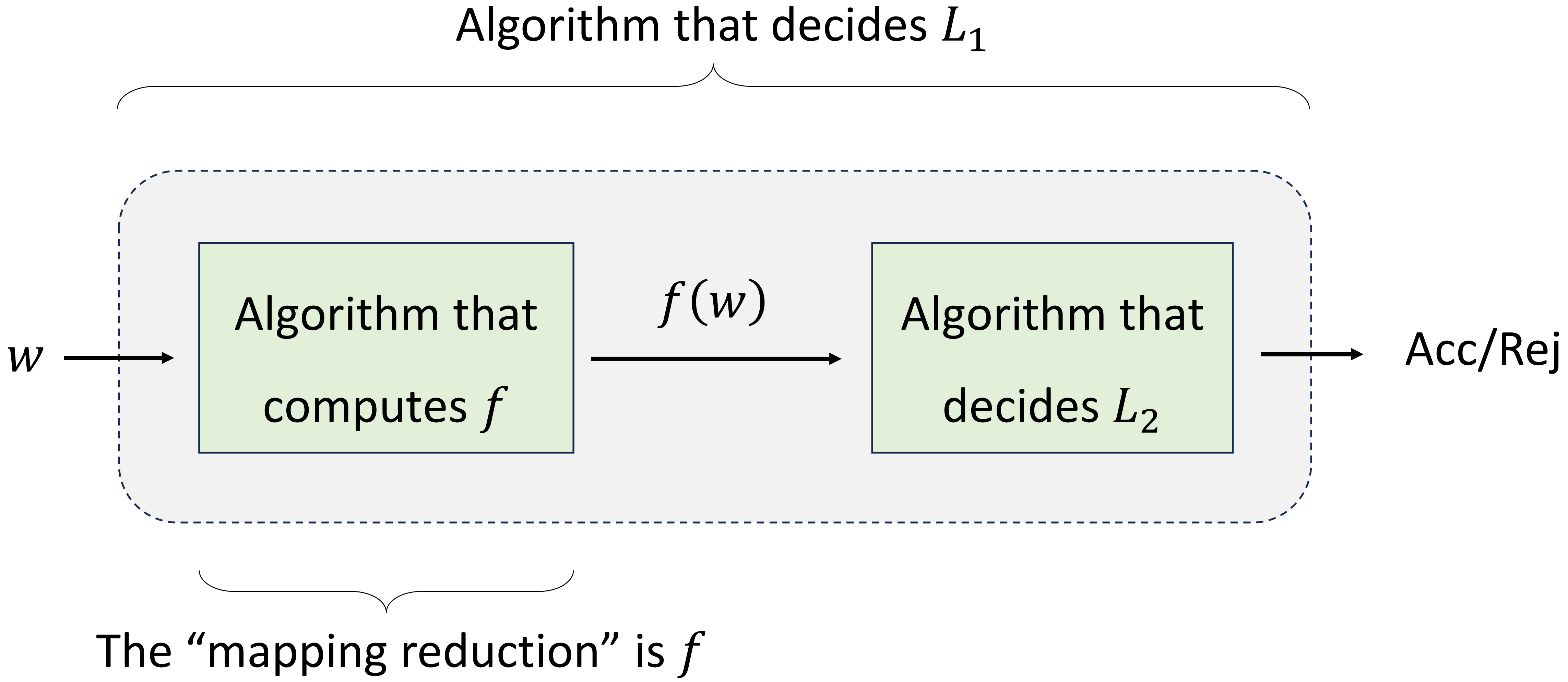
- Informally, a mapping reduction from L_1 to L_2 is a way of converting instances of L_1 into equivalent instances of L_2
  - Note: Any string $w \in \Sigma^*$ is called an "instance" of $L \subseteq \Sigma^*$



Using reductions to prove decidability

- Suppose there exists a mapping reduction f from L_1 to L_2
- **Claim:** If L_2 is decidable, then L_1 is decidable
- **Proof:** Given $w \in \Sigma_1^*$:
  1. Compute $f(w) \in \Sigma_2^*$ (this is possible because f is computable)
  2. Check whether $f(w) \in L_2$ (this is possible because L_2 is decidable)
  3. Accept if $f(w) \in L_2$ and reject if $f(w) \notin L_2$

Using reductions to prove decidability



Using reductions to prove undecidability

- Suppose there exists a mapping reduction f from L_1 to L_2
- **Claim:** If L_1 is undecidable, then L_2 is undecidable
- **Proof:** If L_2 were decidable, then L_1 would be decidable

Using reductions to prove undecidability

- Strategy for proving that some language L is undecidable:
  - Identify a suitable language L_{HARD} that we previously proved is undecidable
  - Design a mapping reduction f from L_{HARD} to L
  - ⚠ *Make sure you do the reduction in the correct direction!*
- The amazing thing about this strategy is that the existence of one algorithm implies the nonexistence of another!

The "emptiness p

Given $\langle M, w\rangle$, how would we compute $f(\langle M, w\rangle)$?

A: Simulate M on w, and if it ever halts, accept

B: Simulate M on w and construct $\langle M'\rangle$ based on simulation results

C: Modify the transition function of M to construct $\langle M'\rangle$

D: There does not exist an algorithm that computes f

Respond at PollEv.com/whoza or text "whoza" to 22333

- Let $\mathrm{E_{TM}} = \{\langle M\rangle : \text{there do}$
- **Claim:** $\mathrm{E_{TM}}$ is undecidable
- **Proof:** We will design a mapping reduction from $\overline{\mathrm{HALT}}$ to $\mathrm{E_{TM}}$
- Let $f(\langle M, w\rangle) = \langle M'\rangle$, where M' is a TM that does the following on input x:
    1. Simulate M on w
    2. If M ever halts, accept
- YES maps to YES ✓ NO maps to NO ✓ Computable ✓