CS21
Decidability and Tractability

Lecture 11

January 31, 2018

Outline

- reductions
- many-one reductions

- undecidable problems
  - computation histories
  - surprising contrasts between decidable/undecidable

Reductions

- Given a new problem NEW, want to determine if it is easy or hard
  - right now, easy typically means decidable
  - right now, hard typically means undecidable
- One option:
  - prove from scratch that the problem is decidable, or
  - prove from scratch that the problem is undecidable (dream up a diag. argument)

Reductions

- A better option:
  - to prove NEW is decidable, show how to transform it into a known decidable problem OLD so that solution to OLD can be used to solve NEW.
  - to prove NEW is undecidable, show how to transform a known undecidable problem OLD into NEW so that solution to NEW can be used to solve OLD.
- called a **reduction**

Example reduction

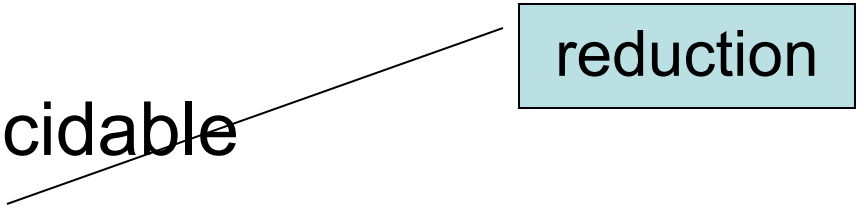
- Try to prove undecidable:

  A_{TM} = {<M, w> : M accepts input w}

- We know this language is undecidable:

  HALT = {<M, w> : M halts on input w}

- Idea:
  - suppose A_{TM} is decidable
  - show that we can use A_{TM} to decide HALT
  - conclude HALT is decidable. Contradiction.



Example reduction

- How could we use procedure that decides A_{TM} to decide HALT?
  - given input to HALT: <M, w>

- Some things we can do:
  - check if <M, w> $\in A_{TM}$
  - construct another TM M' and check if <M', w> $\in A_{TM}$

Example reduction

- Deciding HALT using a procedure that decides A_{TM} ("reducing HALT to A_{TM}").
  - on input <M, w>
  - check if <M, w> $\in A_{TM}$
    - if yes, the M halts on w; **ACCEPT**
    - if no, then M either rejects w or it loops on w
  - construct M' by swapping q_{accept}/q_{reject} in M
  - check if <M', w> $\in A_{TM}$
    - if yes, then M' accepts w, so M rejects w; **ACCEPT**
    - if no, then M neither accepts nor rejects w; **REJECT**

Example reduction

- Preceding reduction proved:

Theorem: A_{TM} is undecidable.

Proof (recap):

- suppose A_{TM} is decidable
- we showed how to use A_{TM} to decide HALT
- conclude HALT is decidable. Contradiction.

Another example

- Try to prove undecidable:

$$E_{TM} = \{<M> : L(M) = \emptyset\}$$

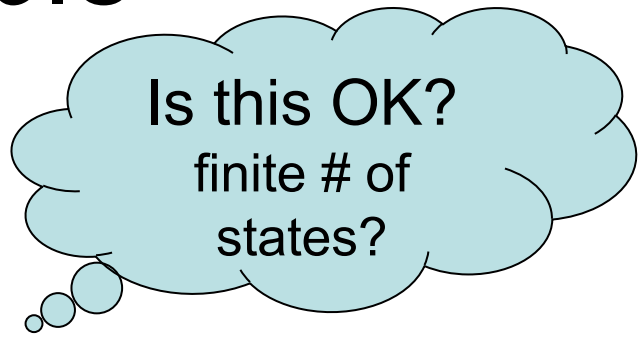
- which problem should we **reduce from**?
  - HALT = {<M, w> : M halts on input w}
  - A_{TM} = {<M, w> : M accepts input w}
- Some things we can do:
  - check if <M> $\in E_{TM}$
  - construct another TM M' and check if <M'> $\in E_{TM}$

Another example

- We are given input <M, w>
- We want to use a procedure that decides E_{TM} to decide if <M, w> $\in A_{TM}$

- Idea:
  - check if <M> $\in E_{TM}$
  - if not?
  - helpful if could make M reject everything except possibly w.

Another example



- Construct TM M':
  - on input x, if x ≠ w, then reject
  - else simulate M on x, and accept if M does.
- on input <M, w>
  - construct M' from description of M
  - check if M' ∈ E_{TM}
    - if no, M must accept w; **ACCEPT**
    - if yes, M cannot accept w; **REJECT**

Another example

- Preceding reduction proved:

Theorem: E_{TM} is undecidable.

Proof (recap):

- suppose E_{TM} is decidable
- we showed how to use E_{TM} to decide A_{TM}
- conclude A_{TM} is decidable. Contradiction.

Definition of reduction

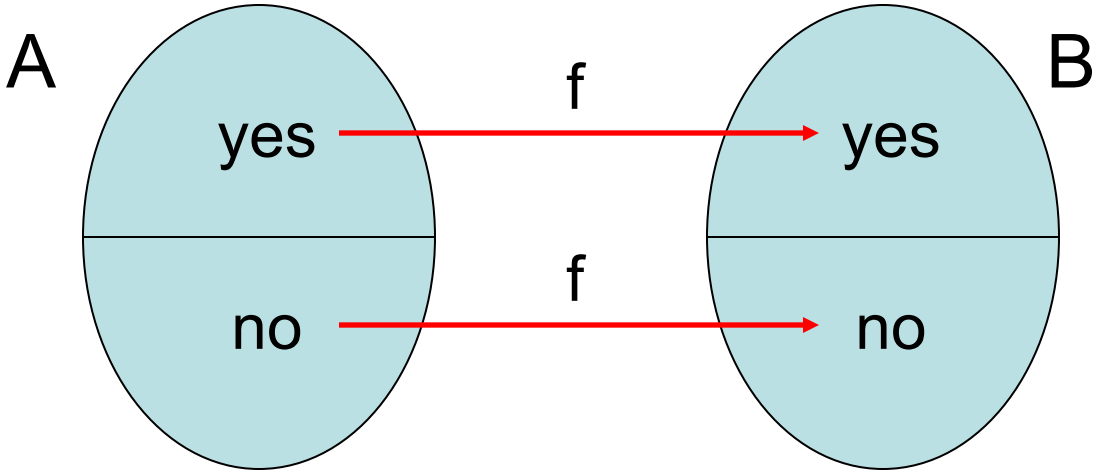
- Can you reduce co-HALT to HALT?

- We know that HALT is RE
- Does this show that co-HALT is RE?
  - recall, we showed co-HALT is not RE

- our current notion of reduction cannot distinguish complements

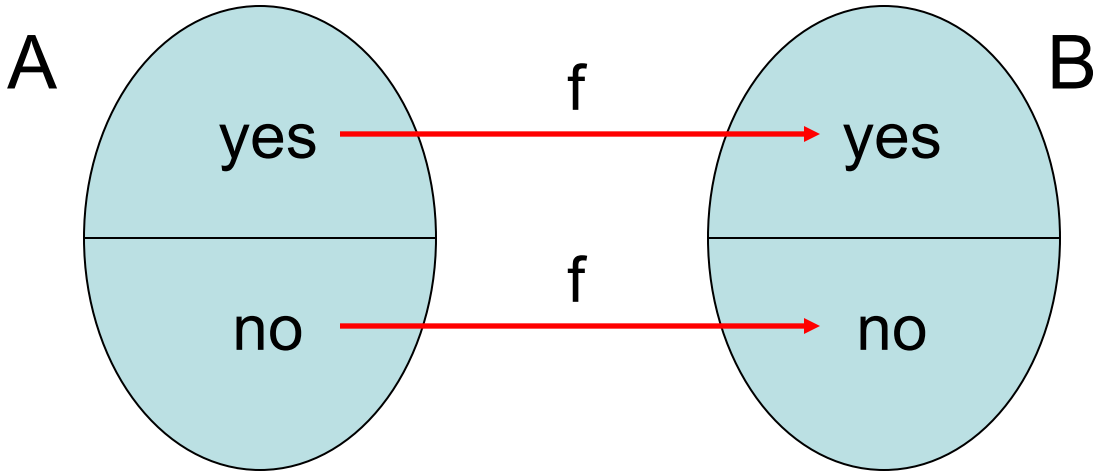
Definition of reduction

- More refined notion of reduction:
  - "many-one" reduction (commonly)
  - "mapping" reduction (book)



reduction from language A to language B

Definition of reduction



- function f should be computable

Definition: $f : \Sigma^* \rightarrow \Sigma^*$ is computable if there exists a TM M_f such that on every $w \in \Sigma^*$ M_f halts on w with f(w) written on its tape.

Definition of reduction

- Notation: “A many-one reduces to B” is written

$$A \leq_m B$$

  – “yes maps to yes and no maps to no” means:

$w \in A$ maps to $f(w) \in B$ & $w \notin A$ maps to $f(w) \notin B$

- B is at least as “hard” as A
  – more accurate: B at least as “expressive” as A

Using reductions

Definition: $A \leq_m B$ if there is a computable function f such that for all w

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

Theorem: if $A \leq_m B$ and B is decidable then A is decidable

Proof:

– decider for A: on input w, compute f(w), run decider for B, do whatever it does.

Using reductions

- Main use: given language NEW, prove it is undecidable by showing OLD $\leq_m$ NEW, where OLD known to be undecidable
  - proof by contradiction
  - if NEW decidable, then OLD decidable
  - OLD undecidable. Contradiction.
- common to reduce in wrong direction.
- review this argument to check yourself.

Using reductions

Theorem: if $A \leq_m B$ and B is RE then A is RE

Proof:

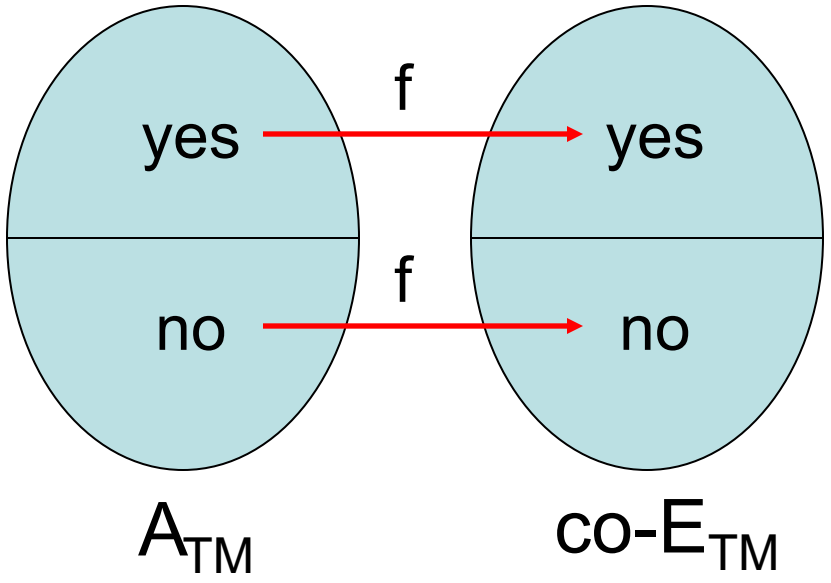
- TM for recognizing A: on input w, compute f(w), run TM that recognizes B, do whatever it does.

- Main use: given language NEW, prove it is not RE by showing OLD $\leq_m$ NEW, where OLD known to be not RE.

Many-one reduction example

- Showed E_{TM} undecidable. Consider:

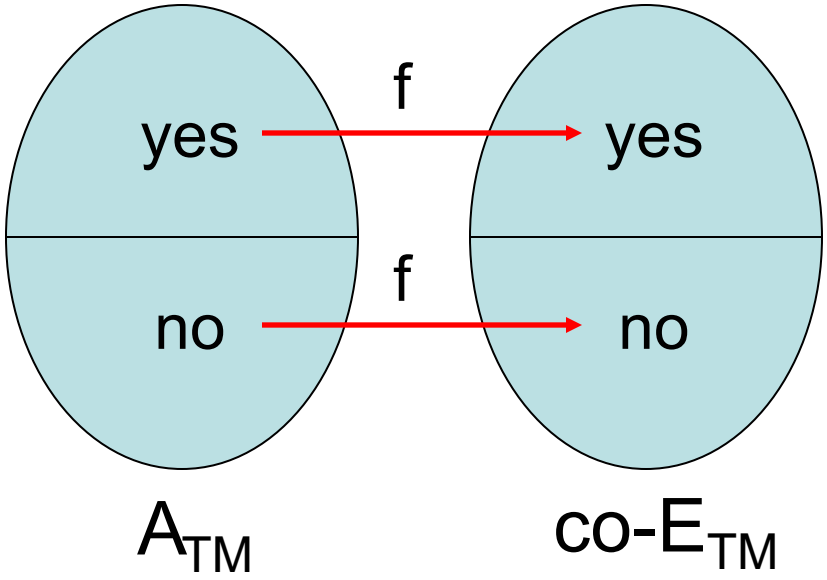
$$co\text{-}E_{TM} = \{<M> : L(M) \neq \emptyset\}$$



- f(<M, w>) = <M'> where M' is TM that
  - on input x, if x ≠ w, then reject
  - else simulate M on x, and accept if M does

- f clearly computable

Many-one reduction example



- f(<M, w>) = <M'>
  where M' is TM that
  - on input x, if x ≠ w, then reject
  - else simulate M on x, and accept if M does

- f clearly computable

- yes maps to yes?
  - if <M, w> ∈ A_{TM} then f(M, w) ∈ co-E_{TM}
- no maps to no?
  - if <M, w> ∉ A_{TM} then f(M, w) ∉ co-E_{TM}

Undecidable problems

Theorem: The language

REGULAR = {<M>: M is a TM and L(M) is regular}

is undecidable.

Proof:

– reduce from A_{TM} (i.e. show $A_{TM} \leq_m$ REGULAR)

– what should f(<M, w>) produce?

Undecidable problems

Proof:

– f(<M, w>) = <M'> described below

on input x:

• if x has form 0^n1^n, accept

• else simulate M on w and accept x if M accepts

• is f computable?

• YES maps to YES?

$\langle M, w\rangle \in A_{TM} \Rightarrow$
$f(M, w) \in$ REGULAR

• NO maps to NO?

$\langle M, w\rangle \notin A_{TM} \Rightarrow$
$f(M, w) \notin$ REGULAR

Dec. and undec. problems

- the boundary between decidability and undecidability is often quite delicate
  - seemingly related problems
  - one decidable
  - other undecidable

- We will see two examples of this phenomenon next.

Computation histories

- Recall configuration of a TM: string uqv with u,v $\in \Gamma^*$, q $\in$ Q
- The sequence of configurations M goes through on input w is a computation history of M on input w
  - may be *accepting*, or *rejecting*
  - reserve the term for halting computations
  - nondeterministic machines may have several computation histories for a given input.

Linear Bounded Automata

LBA definition: TM that is prohibited from moving head off right side of input.

- machine prevents such a move, just like a TM prevents a move off left of tape

- How many possible configurations for a LBA M on input w with |w| = n, m states, and p = $|\Gamma|$?
  - counting gives: mnp^n

Dec. and undec. problems

- two problems we have seen with respect to TMs, now regarding LBAs:
  - LBA acceptance:

    A_{LBA} = {<M, w> : LBA M accepts input w}
  - LBA emptiness:

    E_{LBA} = {<M> : LBA M has L(M) = Ø}
- Both decidable? both undecidable? one decidable?