CS21
Decidability and Tractability
Lecture 12
February 3, 2017
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
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 \}
  – \( A_{TM} = \{ <M, w> : M \text{ accepts input } w \} \)

• 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 $x \neq w$, then reject
  – else simulate $M$ on $x$, and accept if $M$ does.

• on input $<M, w>$
  – construct $M'$ from description of $M$
  – check if $M' \in E_{TM}$
    • if no, $M$ must accept $w$; **ACCEPT**
    • if yes, $M$ cannot accept $w$; **REJECT**

Is this OK? finite # of states?
Another example

• Preceding reduction proved:

**Theorem**: $E_{\text{TM}}$ is undecidable.

**Proof (recap)**:

– suppose $E_{\text{TM}}$ is decidable
– we showed how to use $E_{\text{TM}}$ to decide $A_{\text{TM}}$
– conclude $A_{\text{TM}}$ 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)

A

\[
\begin{array}{c}
\text{yes} \\
\text{no}
\end{array}
\]

B

\[
\begin{array}{c}
\text{yes} \\
\text{no}
\end{array}
\]

\text{reduction from language A to language B}

\text{function } f
Definition of reduction

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.

- function \( f \) should be computable

![Diagram showing reduction relationship between sets A and B]

- for every yes,input to A, \( f \) maps to yes,input to B
- for every no,input to A, \( f \) maps to no,input to B
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:
\[ w \in A \text{ maps to } f(w) \in B \text{ & } w \notin A \text{ maps to } f(w) \notin B \]

• 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 \( w \)
\[ w \in A \iff f(w) \in B \]

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:

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

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

- $f$ clearly computable
Many-one reduction example

• yes maps to yes?
  – if \(<M, w> \in A_{TM}\) then \(f(M, w) \in co-E_{TM}\)

• no maps to no?
  – if \(<M, w> \notin A_{TM}\) then \(f(M, w) \notin co-E_{TM}\)

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

\(f\) clearly computable
Undecidable problems

**Theorem**: The language
\[
\text{REGULAR} = \{<M>: M \text{ is a TM and } L(M) \text{ is regular}\}
\]
is undecidable.

**Proof**:

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

$<M, w> \in A_{TM} \Rightarrow f(M, w) \in \text{REGULAR}$

• NO maps to NO?

$<M, w> \notin A_{TM} \Rightarrow f(M, w) \notin \text{REGULAR}$
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 = |Γ| ?
  - counting gives: mnp^n
Dec. and undec. problems

• two problems we have seen with respect to TMs, now regarding LBAs:
  – LBA acceptance:
    \[ A_{\text{LBA}} = \{ <M, w> : \text{LBA } M \text{ accepts input } w \} \]
  – LBA emptiness:
    \[ E_{\text{LBA}} = \{ <M> : \text{LBA } M \text{ has } L(M) = \emptyset \} \]
• Both decidable? both undecidable? one decidable?
Dec. and undec. problems

**Theorem:** \( A_{LBA} \) is decidable.

**Proof:**
- input \(<M, w>\) where \(M\) is a LBA
- key: only \(mnp^n\) configurations
- if \(M\) hasn’t halted after this many steps, it must be looping forever.
- simulate \(M\) for \(mnp^n\) steps
- if it halts, accept or reject accordingly,
- else reject since it must be looping
Dec. and undec. problems

**Theorem**: $E_{LBA}$ is undecidable.

**Proof**: 
- reduce from $\text{co-}A_{TM}$ (i.e. show $\text{co-}A_{TM} \leq_m E_{LBA}$) 
- what should $f(<M, w>)$ produce? 
- Idea: 
  - produce LBA $B$ that accepts exactly the accepting computation histories of $M$ on input $w$
Dec. and undec. problems

Proof:

– $f(<M, w>) = <B>$ described below

on input $x$, check if $x$ has form

$\#C_1\#C_2\#C_3\#\ldots\#C_k\#$

• check that $C_1$ is the start configuration for $M$ on input $w$

• check that $C_i \Rightarrow^1 C_{i+1}$

• check that $C_k$ is an accepting configuration for $M$

• is $B$ an LBA?

• is $f$ computable?

• YES maps to YES?

$<M, w> \in \text{co-}A_{\text{TM}} \Rightarrow f(M, w) \in E_{\text{LBA}}$

• NO maps to NO?

$<M, w> \notin \text{co-}A_{\text{TM}} \Rightarrow f(M, w) \notin E_{\text{LBA}}$