CS21
Decidability and Tractability

Lecture 14
February 4, 2022

Outline

• many-one reductions
• undecidable problems
  – computation histories
  – surprising contrasts between decidable/undecidable
• Rice’s Theorem
• Post Correspondence Problem

Definition of reduction

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

A

f

yes

no

B

f

yes

no

reduction from language A to language B

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 ≤_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

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: $A_{LBA} = \{<M, w> : \text{LBA } M \text{ accepts input } w\}$
  - LBA emptiness: $E_{LBA} = \{<M> : \text{LBA } M \text{ has } L(M) = \emptyset\}$
- 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 co-$A_{TM}$ (i.e. show 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 $\#C_1\#C_2\#C_3\#...\#C_k\#$
  - check that $C_1$ is the start configuration for $M$ on input $w$
  - check that $C_k \Rightarrow C_{k+1}$
  - check that $C_k$ is an accepting configuration for $M$

  - is $B$ an LBA?
  - is $f$ computable?
  - YES maps to YES? $<M, w> \in \text{co-}A_{TM} \Rightarrow f(M, w) \in E_{LBA}$
  - NO maps to NO? $<M, w> \in \text{co-}A_{TM} \Rightarrow f(M, w) \notin E_{LBA}$
Dec. and undec. problems

- two problems regarding Context-Free Grammars:
  - does a CFG generate all strings:
    \( \text{ALL}_{\text{CFG}} = \{ \langle G \rangle : G \text{ is a CFG and } L(G) = \Sigma^* \} \)
  - CFG emptiness:
    \( \text{E}_{\text{CFG}} = \{ \langle G \rangle : G \text{ is a CFG and } L(G) = \emptyset \} \)

- Both decidable? both undecidable? one decidable?

---

**Theorem:** \( \text{E}_{\text{CFG}} \) is decidable.

**Proof:**
- observation: for each nonterminal \( A \), the set \( S_A = \{ w : A \Rightarrow^* w \} \) is non-empty iff there is some rule: \( A \to x \) and for all non-terminals \( B \) in string \( x \), \( S_B \neq \emptyset \)
- on input \( \langle G \rangle \)
  - mark all terminals in \( G \)
  - repeat until no new non-terminals get marked:
    - if there is a production \( A \to x_1x_2x_3 \ldots x_k \)
      - and each symbol \( x_1, x_2, \ldots, x_k \) has been marked
      - then mark \( A \)
    - if \( S \) marked, reject (\( G \notin \text{E}_{\text{CFG}} \)), else accept (\( G \in \text{E}_{\text{CFG}} \)).
- terminates? correct?

---

**Theorem:** \( \text{ALL}_{\text{CFG}} \) is undecidable.

**Proof:**
- reduce from \( \text{co-} \text{A}_{\text{TM}} \) (i.e. show \( \text{co-} \text{A}_{\text{TM}} \leq_m \text{ALL}_{\text{CFG}} \))
- what should \( f(<M, w>) \) produce?
- Idea:
  - produce CFG \( G \) that generates all strings that are not accepting computation histories of \( M \) on \( w \)

---

Dec. and undec. problems

**Proof:**
- build a NPDA, then convert to CFG
- want to accept strings not of this form,

\[ \#C_1\#C_2\#C_3\# \ldots \#C_n\# \]

plus strings of this form but where
- \( C_1 \) is not the start config. of \( M \) on input \( w \), or
- \( C_n \) is not an accept. config. of \( M \) on input \( w \), or
- \( C_i \) does not yield in one step \( C_{i+1} \) for some \( i \)

---
Dec. and undec. problems

Proof:
- checking:
  - $C_i$ does not yield in one step $C_{i+1}$ for some $i$
  - push $C_i$ onto stack
  - at $\#$, start popping $C_i$ and compare to $C_{i+1}$
    - accept if mismatch away from head location, or
    - symbols around head changed in a way inconsistent with $M$’s transition function.
  - is everything described possible with NPDA?

Dec. and undec. problems

Proof:
- Problem: cannot compare $C_i$ to $C_{i+1}$
- could prove in same way that proved
  $\{ww: w \in \Sigma^*\}$ not context-free
- recall that
  $\{ww^R: w \in \Sigma^*\}$ is context-free
- free to tweak construction of $G$ in the reduction
- solution: write computation history:
  \[
  \#C_1\#C_2\#\#C_3\#\#C_4\#\ldots\#C_k\#
  \]

Dec. and undec. problems

Proof:
- $f(<M, w>) = <G>$ equiv. to NPDA below:
  on input $x$, accept if not of form:
  \[
  \#C_1\#C_2\#\#C_3\#\#C_4\#\ldots\#C_k\#
  \]
  - accept if $C_1$ is the not the start configuration for $M$ on input $w$
  - accept if check that $C_i$ does not yield in one step $C_{i+1}$
  - accept if $C_k$ is not an accepting configuration for $M$
  - is $f$ computable?
  - YES maps to YES?
    - $<M, w> \in \text{co-}\text{A}_{TM} \Rightarrow f(M, w) \in \text{ALL}_{CFG}$
  - NO maps to NO?
    - $<M, w> \in \text{co-}\text{A}_{TM} \Rightarrow f(M, w) \not\in \text{ALL}_{CFG}$