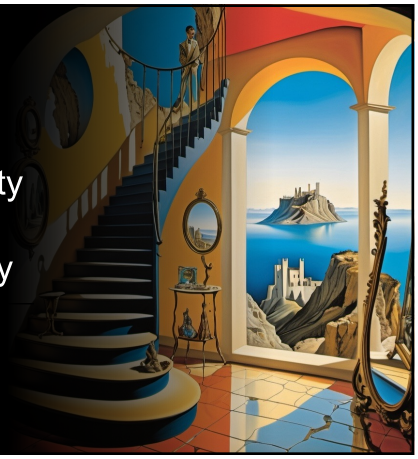
CS21 Decidability and Tractability

Lecture 13
February 5, 2025



1

Outline

- undecidable problems
  - computation histories
  - surprising contrasts between decidable/undecidable
- Rice's Theorem
- Post Correspondence Problem (skip?)
- Beyond RE and co-RE
- Recursion Theorem

2

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.

3

Computation histories

- Recall configuration of a TM: string uqv with u,v ∈ Γ^*, q ∈ 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.

4

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

5

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?

6

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

7

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

8

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 $C_i \Rightarrow^1 C_{i+1}$
- check that C_k is an accepting configuration for M

- is B an LBA?
- is f computable?
- YES maps to YES?

  <M, w> $\in$ co-$A_{TM} \Rightarrow$ f(M, w) $\in E_{LBA}$
- NO maps to NO?

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

9

Dec. and undec. problems

- two problems regarding Context-Free Grammars:
  - does a CFG generate all strings:

    ALL_{CFG} = {<G> : G is a CFG and L(G) = Σ^*}
  - CFG emptiness:

    E_{CFG} = {<G> : G is a CFG and L(G) = Ø}
- Both decidable? both undecidable? one decidable?

10

Dec. and undec. problems

Theorem: E_{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:

  $A \rightarrow \mathbf{x}$

  and for all non-terminals B in string x, $S_B \neq \emptyset$

11

Dec. and undec. problems

Proof:

- on input <G>
- mark all terminals in G
- repeat until no new non-terminals get marked:
  - if there is a production $A \rightarrow x_1x_2x_3\ldots x_k$
  - and each symbol $x_1, x_2, \ldots, x_k$ has been marked
  - then mark A
- if S marked, reject ($G \notin E_{CFG}$), else accept ($G \in E_{CFG}$).
- terminates? correct?

12

Dec. and undec. problems

Theorem: ALL_{CFG} is undecidable.

Proof:

- reduce from co-A_{TM} (i.e. show co-$A_{TM} \leq_m ALL_{CFG}$)
- 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,

  $\#C_1\#C_2\#C_3\#...\#C_k\#$

  plus strings of this form but where
  - C_1 is not the start config. of M on input w, or
  - C_k 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:

- our NPDA nondeterministically checks one of:
  - C_1 is not the start config. of M on input w, or
  - C_k 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
  - input has fewer than two #'s
- details of first two?
- to check third condition:
  - nondeterministically guess C_i starting position
  - how to check that C_i doesn't yield in 1 step C_{i+1} ?

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:

  $\#C_1\#C_2^R\#C_3\#C_4^R...\#C_k\#$

Dec. and undec. problems

Proof:

- f(<M, w>) = <G> equiv. to NPDA below:

on input x, accept if not of form:

$\#C_1\#C_2^R\#C_3\#C_4^R...\#C_k\#$

- 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?

  $\langle M, w\rangle \in$ co-$A_{TM} \Rightarrow f(M, w) \in ALL_{CFG}$
- NO maps to NO?

  $\langle M, w\rangle \notin$ co-$A_{TM} \Rightarrow f(M, w) \notin ALL_{CFG}$