Decidability and Tractability

Lecture 15
February 7, 2022

Outline

• undecidable problems
  – surprising contrasts between decidable/undecidable (finishing up)
• Rice’s Theorem
• Post Correspondence Problem

Dec. and undec. problems

• two problems regarding Context-Free Grammars:
  – does a CFG generate all strings:
    \[ \text{ALL}_{\text{CFG}} = \{<G> : G \text{ is a CFG and } L(G) = \Sigma^*\} \]
  – CFG emptiness:
    \[ \text{E}_{\text{CFG}} = \{<G> : G \text{ is a CFG and } L(G) = \emptyset\} \]
• Both decidable? both undecidable? one decidable?

Dec. and undec. problems

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

**Proof:**

– reduce from \( \text{co-}A_{\text{TM}} \) (i.e. show \( \text{co-}A_{\text{TM}} \leq_m \text{ALL}_{\text{CFG}} \))
– 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,

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

plus strings of this form but where

• \( C_i \) is not the start config. of \( M \) on input \( w \), or
• \( C_i \) 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:
  • \( C_i \) is not the start config. of \( M \) on input \( w \), or
  • \( C_i \) 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 \( \{ \text{ww} \in \Sigma^* \} \) not context-free
- recall that \( \{ \text{ww}^R \in \Sigma^* \} \) is context-free
- free to tweak construction of \( G \) in the reduction
- solution: write computation history:
  \[ \#C_1\#C_2\#C_3\#C_4\ldots\#C_k\# \]

Rice’s Theorem

- We have seen that the following properties of TM’s are undecidable:
  - TM accepts string \( w \)
  - TM halts on input \( w \)
  - TM accepts the empty language
  - TM accepts a regular language
- Can we describe a single generic reduction for all these proofs?
- Yes. Every property of TMs undecidable!

Rice’s Theorem

- A TM property is a language \( P \) for which
  - if \( L(M_1) = L(M_2) \) then \( <M_1> \in P \) iff \( <M_2> \in P \)
- TM property \( P \) is nontrivial if
  - there exists a TM \( M_1 \) for which \( <M_1> \in P \), and
  - there exists a TM \( M_2 \) for which \( <M_2> \notin P \).

Rice’s Theorem: Every nontrivial TM property is undecidable.

Rice’s Theorem

- The setup:
  - let \( T_\emptyset \) be a TM for which \( L(T_\emptyset) = \emptyset \)
    - technicality: if \( <T_\emptyset> \in P \) then work with property co-\( P \) instead of \( P \).
    - conclude co-\( P \) undecidable; therefore \( P \) undec.
      due to closure under complement
  - so, WLOG, assume \( <T_\emptyset> \notin P \)
  - non-triviality ensures existence of TM \( M_1 \) such that \( <M_1> \in P \)
Rice's Theorem

**Proof:**
- reduce from \(A_{TM} \) (i.e. show \(A_{TM} \leq_m P\))
- what should \(f(<M, w>)\) produce?
- \(f(<M, w>) = <M'>\) described below:
  
<table>
<thead>
<tr>
<th>1. f computable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES maps to YES?</td>
</tr>
<tr>
<td>(intersection of two RE languages)</td>
</tr>
</tbody>
</table>

Post Correspondence Problem

- many undecidable problems unrelated to TMs and automata

- classic example: Post Correspondence Problem
  
  \(PCP = \{<(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)> : x_i, y_i \in \Sigma^* \text{ and there exists } (a_1, a_2, \ldots, a_n) \text{ for which } x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_n \}\)

Post Correspondence Problem

**Theorem:** PCP is undecidable.

Proof:
- reduce from \(A_{TM} \) (i.e. show \(A_{TM} \leq_m P\))
- two step reduction makes it easier
- first, show \(A_{TM} \leq_m \text{MPCP}\)
  
  \[(\text{MPCP} = "\text{modified PCP}"")\
- next, show \(\text{MPCP} \leq_m \text{PCP}\)

Post Correspondence Problem

\(\text{PCP} = \{<(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)> : x_i, y_i \in \Sigma^* \text{ and there exists } (a_1, a_2, \ldots, a_n) \text{ for which } x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_n \}\)

Proof of \(\text{MPCP} \leq_m \text{PCP}\):
- notation: for a string \(u = u_1 u_2 u_3 \ldots u_n\)
  
  \* \(u^*\) means the string \(u_1 u_2 u_3 \ldots u_n\)
  
  \* \(u^+\) means the string \(u_1 u_2 u_3 \ldots u_n^*\)
  
  \* \(u^*\) means the string \(u_1 u_2 u_3 \ldots u_n^*\)
Proof of \( \text{MPCP} \leq_m \text{PCP} \):

- given an instance \((x_1, y_1), \ldots, (x_n, y_n)\) of \( \text{MPCP} \)
- produce an instance of \( \text{PCP} \): \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
- YES maps to YES?
  - given a match in original \( \text{MPCP} \) instance, can produce a match in new \( \text{PCP} \) instance
- NO maps to NO?
  - given a match in new \( \text{PCP} \) instance, can produce a match in original \( \text{MPCP} \) instance

Post Correspondence Problem

Theorem: \( \text{PCP} \) is undecidable.

Proof:

- show \( A_{TM} \leq_m \text{MPCP} \)

\[ MPCP = \{ <(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)> \mid x_i, y_i \in \Sigma^* \text{ and there exists } (a_1, a_2, \ldots, a_n) \text{ for which } x_1 x_2 \cdots x_n = y_1 y_2 \cdots y_n \} \]

- show \( \text{MPCP} \leq_m \text{PCP} \)

Proof of \( A_{TM} \leq_m \text{MPCP} \):

- given instance of \( A_{TM}; <M, w> \)
- idea: a match will record an accepting computation history for \( M \) on input \( w \)
- start tile records starting configuration:
  - add tile \((\#_1, \#_2, \ldots, \#_n, \#_w, \#_t)\)

\[
\begin{array}{cccc}
\# & \#_1 & \#_2 & \#_n \\
\#_w & \#_t & \#_q & \#_r \\
\end{array}
\]
Post Correspondence Problem

- tiles for copying (not near head)
  - for all \( a \in \Gamma \), add tile \((a, a)\)
- tiles for copying # marker
  - add tile \((#, #)\)
- tiles for copying # marker and adding \( _ \) to end of tape
  - add tile \((#, _#)\)

Post Correspondence Problem

- tiles for deleting symbols to left of \( q_{\text{accept}} \)
  - for all \( a \in \Gamma \), add tile \((a_{\text{accept}}, q_{\text{accept}})\)

Post Correspondence Problem

- tiles for deleting symbols to right of \( q_{\text{accept}} \)
  - for all \( a \in \Gamma \), add tile \((a_{\text{accept}}, a_{\text{accept}})\)

Post Correspondence Problem

- YES maps to YES?
  - by construction, if \( M \) accepts \( w \), there is a way to assemble the tiles to achieve this match:
    
    \[
    \begin{array}{cccccccc}
    \# & \#a_{\text{accept}}v_1 & \#a_{\text{accept}}v_2 & \ldots & \#a_{\text{accept}}v_m & \#C_1 & \#C_2 & \ldots & \#C_m & \#
    \end{array}
    \]
    
    \( \text{where } C_1, C_2, \ldots, C_m \text{ is an accepting computation history} \)

- NO maps to NO?
  - sketch: at any step if the “intended” next tile is not used, then it is impossible to recover and produce a match in the end (case analysis)

We have proved:

**Theorem:** PCP is undecidable.

by showing:

- \( A_{\text{TM}} \leq_m \text{MPCP} \)
- \( \text{MPCP} \leq_m \text{PCP} \)
- conclude \( A_{\text{TM}} \leq_m \text{PCP} \)