Undecidable Problems for CFGs

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN
Fall 2018

Chinese University of Hong Kong
Decidable vs undecidable

<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFA $D$ accepts $w$</td>
<td>TM $M$ accepts $w$</td>
</tr>
<tr>
<td>CFG $G$ generates $w$</td>
<td>TM $M$ halts on $w$</td>
</tr>
<tr>
<td>DFAs $D$ and $D'$ accept same inputs</td>
<td>TM $M$ accepts some input</td>
</tr>
<tr>
<td></td>
<td>TM $M$ and $M'$ accept the same inputs</td>
</tr>
</tbody>
</table>

CFG $G$ generates all inputs?

CFG $G$ is ambiguous?
Representing computation

$L_1 = \{w\%w \mid w \in \{a, b\}^*\}$
A configuration consists of current state, head position, and tape contents.

Configuration (abbreviation)

- ab $q_1$ a
- abb $q_{acc}$
If $M$ halts on $w$, the **computation history** of $(M, w)$ is the sequence of configurations $C_1, \ldots, C_k$ that $M$ goes through on input $w$.

The computation history can be written as a string $h$ over alphabet $\Gamma \cup Q \cup \{\#\}$

### Accepting History
- $M$ accepts $w$ $\iff$ $q_{\text{acc}}$ appears in $h$

### Rejecting History
- $M$ rejects $w$ $\iff$ $q_{\text{rej}}$ appears in $h$
Undecidable problems for CFGs

\[ \text{ALL}_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG that generates all strings} \} \]

The language \( \text{ALL}_{\text{CFG}} \) is undecidable

We will argue that

If \( \text{ALL}_{\text{CFG}} \) can be decided, so can \( \overline{A_{TM}} \)

\[ \overline{A_{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that rejects or loops on } w \} \]
Undecidable problems for CFGs

Proof by contradiction

Suppose some Turing machine $A$ decides $\text{ALL}_{\text{CFG}}$

$\langle G \rangle \xrightarrow{A} \text{accept if } G \text{ generates all strings}$

$\langle G \rangle \xrightarrow{A} \text{reject otherwise}$

We want to construct a Turing machine $S$ that decides $\overline{A_{TM}}$

$\langle M, w \rangle \xrightarrow{\text{Convert to } G} \langle G \rangle \xrightarrow{A} \text{accept if } M \text{ rejects or loops on } w$

$\langle G \rangle \xrightarrow{A} \text{reject if } M \text{ accepts } w$

$G$ generates all strings if $M$ rejects or loops on $w$

$G$ fails to generate some string if $M$ accepts $w$
Undecidable problems for CFGs

\[ \langle M, w \rangle \xrightarrow{\text{Convert to } G} \langle G \rangle \]

\( G \) fails to generate some string

\[ \iff \]

\( M \) accepts \( w \)

The alphabet of \( G \) will be \( \Gamma \cup Q \cup \{\#\} \)

\( G \) will generate all strings except

accepting computation history of \( (M, w) \)

First we construct a PDA \( P \), then convert it to CFG \( G \)
Undecidability via computation histories

candidate computation history \( h \) of \((M, w)\)

\[
\begin{array}{c}
\# q_0 a b \# a x q_1 b \# a b \# \ldots \# x x x \# q_{acc} x \# \\ \Rightarrow \text{Reject}
\end{array}
\]

\( P = \) on input \( h \) (try to spot a mistake in \( h \))

- If \( h \) is not of the form \( \# w_1 \# w_2 \# \ldots \# w_k \# \), accept
- If \( w_1 \neq q_0 w \) or \( w_k \) does not contain \( q_{acc} \), accept
- If two consecutive blocks \( w_i \# w_{i+1} \) do not follow from the transitions of \( M \), accept

Otherwise, \( h \) must be an accepting history, reject
Computation is local

Changes between configurations always occur around the head
Legal and illegal transitions windows

legal windows

... abx ...
... abx ...
... a$q_3$a ...
... q$_6$ax ...
... aba ...
... ab$q_6$ ...
... aa□ ...
... xa□ ...

illegal windows

... q$_3$ab ...
... ab$q_3$ ...
... q$_3$q$_3$a ...
... q$_3$q$_3$x ...
... a$q_3$a ...
... q$_6$ab ...
... a$q_3$a ...
... a$q_6$x ...
Implementing $P$

If two consecutive blocks $w_i \# w_{i+1}$ do not follow from the transitions of $M$, accept

For every position of $w_i$:
- Remember offset from $\#$ in $w_i$ on stack
- Remember first row of window in state

After reaching the next $\#$:
- Pop offset from $\#$ from stack as you consume input
- Remember second row of window in state

If window is illegal, accept; Otherwise reject
The computation history method

\[ \text{ALL}_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG that generates all strings} \} \]

- If \text{ALL}_{\text{CFG}} can be decided, so can \overline{\text{A}_{\text{TM}}}

\[ \langle M, w \rangle \xrightarrow{\text{Convert to } G} \langle G \rangle \]

- \( G \) accepts all strings except accepting computation history of \((M, w)\)

We first construct a PDA \( P \), then convert it to CFG \( G' \)
Post Correspondence Problem

Input: A fixed set of tiles, each containing a pair of strings

\[
\begin{array}{cccccc}
\text{bab} & \text{c} & \text{a} & \text{baa} & \text{a} & \text{bab} \\
\text{cc} & \text{ab} & \text{ab} & \text{a} & \text{baba} & \text{ε} \\
\end{array}
\]

Given an infinite supply of tiles from a particular set, can you match top and bottom?

\[
\begin{array}{cccccccc}
\text{a} & \text{baa} & \text{bab} & \text{ε} & \text{c} & \text{c} & \text{bab} & \text{a} \\
\text{ab} & \text{a} & \text{ε} & \text{ab} & \text{ab} & \text{cc} & \text{baba} & \\
\end{array}
\]

Top and bottom are both abaababcccbaba
PCP = \{ \langle T \rangle \mid T \text{ is a collection of tiles that contains a top-bottom match} \}

Next lecture we will show (using computation history method)

The language PCP is undecidable
Ambiguity of CFGs

\[ \text{AMB} = \{ \langle G \rangle \mid G \text{ is an ambiguous CFG} \} \]

The language AMB is undecidable

We will argue that

If AMB can be decided, then so can PCP
Ambiguity of CFGs

$T$ (collection of tiles) $\mapsto G$ (CFG)

If $T$ can be matched, then $G$ is ambiguous
If $T$ cannot be matched, then $G$ is unambiguous

First, let’s number the tiles

1. bab cc
2. c ab
3. a ab
Ambiguity of CFGs

\[ T \text{ (collection of tiles)} \quad \rightleftharpoons \quad G \text{ (CFG)} \]

1. bab
2. c
3. a

Terminals: a, b, c, 1, 2, 3

Variables: S, T, B

Productions:

- \[ S \rightarrow T \mid B \]
- \[ T \rightarrow \text{bab}T1 \]
- \[ T \rightarrow \text{bab}1 \]
- \[ B \rightarrow \text{cc}B1 \]
- \[ B \rightarrow \text{cc}1 \]
- \[ T \rightarrow \text{c}T2 \]
- \[ T \rightarrow \text{c}2 \]
- \[ B \rightarrow \text{ab}B2 \]
- \[ B \rightarrow \text{ab}2 \]
- \[ T \rightarrow \text{a}T3 \]
- \[ T \rightarrow \text{a}3 \]
- \[ B \rightarrow \text{ab}B3 \]
- \[ B \rightarrow \text{ab}3 \]
Each sequence of tiles gives a pair of derivations:

\[ S \Rightarrow T \Rightarrow babT_1 \Rightarrow babcT_21 \Rightarrow babcc221 \]
\[ S \Rightarrow B \Rightarrow ccB_1 \Rightarrow ccabB_21 \Rightarrow ccabab221 \]

If the tiles match, these two derive the same string (with different parse trees)
Ambiguity of CFGs

\[ T \text{ (collection of tiles)} \quad \rightarrow \quad G \text{ (CFG)} \]

If \( T \) can be matched, then \( G \) is ambiguous

If \( T \) cannot be matched, then \( G \) is unambiguous

If \( G \) is ambiguous, then the two parse trees will look like

Therefore \( n_1 n_2 \ldots n_i = m_1 m_2 \ldots m_j \), and there is a match