Undecidable Problems for CFGs
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2017
## 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)

$ab \ q_1 \ a$

$abb \ q_{acc}$
Computation histories

\[ q^0 \ abb%abb \]
\[ x \ q^2 \ bb%abb \]
\[ \vdots \]
\[ \text{xbb} \ q_2 \ %abb \]
\[ \text{xbb} \ q_3 \ \abb \]
\[ \vdots \]
\[ \text{xxx%xxx} \ q_1 \]
\[ \text{xxx%xx} \ q_{acc} \ x \]

computation history
Computation histories as strings

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 $w$ $\iff$ $q_{acc}$ appears in $h$

rejecting history: $M$ rejects $w$ $\iff$ $q_{rej}$ 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 \( \overline{A_{\text{TM}}} \)

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

Proof by contradiction

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

$\langle G \rangle \rightarrow A \rightarrow \text{accept if } G \text{ generates all strings}$
$\rightarrow \text{reject otherwise}$

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

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

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

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

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

\[ G \text{ fails to generate some string} \]
\[ \Updownarrow \]
\[ M \text{ accepts } w \]

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

\( G \) will generate all strings except accepting computation histories of \( (M, w) \)

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

candidate computation history $h$ of $(M, w)$ → $P$ → accept everything except accepting $h$

$\#q_0ab%ab#xq_1b%ab#…#xx%xq_{acc}x#$ ⇒ Reject

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

- If $h$ is not of the form $\#w_1\#w_2\#…\#w_k#$, accept
- If $w_1 \neq q_0w$ 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

<table>
<thead>
<tr>
<th>legal windows</th>
<th>illegal windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>... abx ...</td>
<td>... q₃ab ...</td>
</tr>
<tr>
<td>... abx ...</td>
<td>... abq₃ ...</td>
</tr>
<tr>
<td>... a_q₃a ...</td>
<td>... q₃q₃a ...</td>
</tr>
<tr>
<td>... q₆ax ...</td>
<td>... q₃q₆x ...</td>
</tr>
<tr>
<td>... aba ...</td>
<td>... aq₃a ...</td>
</tr>
<tr>
<td>... abq₆ ...</td>
<td>... q₆ab ...</td>
</tr>
<tr>
<td>... aa□ ...</td>
<td>... aq₆x ...</td>
</tr>
<tr>
<td>... xa□ ...</td>
<td>... aq₆x ...</td>
</tr>
</tbody>
</table>
Implementing $P$

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

\[
\text{For every position of } w_i:\n\begin{align*}
&\text{Remember offset from } \# \text{ in } w_i \text{ on stack} \\
&\text{Remember first row of window in state} \\
&\text{After reaching the next } \#: \\
&\text{Pop offset from } \# \text{ from stack as you consume input} \\
&\text{Remember second row of window in state} \\
&\text{If window is illegal, accept; Otherwise reject}
\end{align*}
\]
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 $A_{\text{TM}}$

$$\langle M, w \rangle \xrightarrow{\text{Convert to } G} \langle G' \rangle$$

$G$ accepts all strings except accepting computation histories 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

<table>
<thead>
<tr>
<th>bab</th>
<th>c</th>
<th>a</th>
<th>baa</th>
<th>a</th>
<th>bab</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc</td>
<td>ab</td>
<td>ab</td>
<td>a</td>
<td>baba</td>
<td>ε</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>a</th>
<th>baa</th>
<th>bab</th>
<th>c</th>
<th>c</th>
<th>bab</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>ab</td>
<td>a</td>
<td>ε</td>
<td>ab</td>
<td>ab</td>
<td>cc</td>
<td>baba</td>
</tr>
</tbody>
</table>

Top and bottom are both abaababcccbaba
Undecidability of PCP

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

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

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

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

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

Variables: \( S, T, B \)

Productions:

\[
S \rightarrow T | B \\
T \rightarrow bab T_1 \quad T \rightarrow c T_2 \quad T \rightarrow a T_3 \\
B \rightarrow cc B_1 \quad B \rightarrow ab B_2 \quad B \rightarrow ab B_3 \\
T \rightarrow bab_1 \quad T \rightarrow c_2 \quad T \rightarrow a_3 \\
B \rightarrow cc_1 \quad B \rightarrow ab_2 \quad B \rightarrow ab_3
\]
Ambiguity of CFGs

Each sequence of tiles gives a pair of derivations

\[
\begin{align*}
S & \Rightarrow T \Rightarrow bab T1 \Rightarrow babc T21 \Rightarrow babcc221 \\
S & \Rightarrow B \Rightarrow cc B1 \Rightarrow ccab B21 \Rightarrow ccabab221
\end{align*}
\]

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

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

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

\[
\begin{align*}
S & \quad S \\
| & | \\
T & B \\
| & | \\
\vdots & \vdots \\
T & B \\
| & | \\
a_1 & b_1 \\
a_2 & b_2 \\
\vdots & \vdots \\
a_i & b_j \\
n_1 & m_1 \\
n_2 & m_2 \\
n_i & m_j \\
\end{align*}
\]

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