Undecidable Problems for CFGs
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2015
<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 computations

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

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>□</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>□</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>□</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td></td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>□</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>b</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>
```

Configuration (abbreviation)

ab $q_1$ a

abb $q_{acc}$
Computation histories

- $q_0$ abb%abb
- a $q_2$ bb%abb
- : ...
- abb $q_2$ %abb
- abb% $q_3$ abb
- abb $q_2$ %xbb
- : ...
- xxx%xxx $q_1$
- xxx%xx $q_a$ 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 $w$ $\iff$ $q_{\text{acc}}$ appears in $h$

rejecting history: $M$ rejects $w$ $\iff$ $q_{\text{rej}}$ appears in $h$
Undecidable problems for CFGs

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

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

We will argue that

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

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

Proof by contradiction

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

$\langle G \rangle \rightarrow A$ accept if $G$ generates all strings

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

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

reject if $M$ 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

$\uparrow$

$M$ 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 $(M, w)$

$P = \text{on input } h \quad \text{(try to spot a mistake in } h\text{)}$

- 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 $q_a$, 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

$\Rightarrow \text{Reject}$

$\# q_0 a b \# x q_1 b \# a b \# \ldots \# x x \% x q_a \# x$
Computation is local

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

Legal windows

⋯ abx ⋯
⋯ abx ⋯
⋯ aq₃a ⋯
⋯ q₆ax ⋯
⋯ aba ⋯
⋯ abq₆ ⋯
⋯ aa □ ⋯
⋯ xa □ ⋯

Illegal windows

⋯ q₃ab ⋯
⋯ abq₃ ⋯
⋯ q₃q₆a ⋯
⋯ q₃q₆x ⋯
⋯ aq₃a ⋯
⋯ q₆ab ⋯
⋯ aq₃a ⋯
⋯ aq₆x ⋯

\[ q₃ \quad a/xL \quad q₆ \]
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 \( \overline{A_{\text{TM}}} \)

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

\( G \) 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

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

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

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

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} \}

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

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

\begin{itemize}
  \item \text{1} \quad \text{bab}
  \item \text{2} \quad \text{c}
  \item \text{3} \quad \text{a}
\end{itemize}

\begin{itemize}
  \item \text{1} \quad \text{cc}
  \item \text{2} \quad \text{ab}
  \item \text{3} \quad \text{ab}
\end{itemize}

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

Variables: S, T, B

Productions:

- \( S \rightarrow T \mid B \)
- \( T \rightarrow \text{bab}T1 \)
- \( T \rightarrow \text{c}T2 \)
- \( T \rightarrow \text{a}T3 \)
- \( B \rightarrow \text{cc}B1 \)
- \( B \rightarrow \text{ab}B2 \)
- \( B \rightarrow \text{ab}B3 \)
- \( T \rightarrow \text{bab}1 \)
- \( T \rightarrow \text{c}2 \)
- \( T \rightarrow \text{a}3 \)
- \( B \rightarrow \text{cc}1 \)
- \( B \rightarrow \text{ab}2 \)
- \( B \rightarrow \text{ab}3 \)
Ambiguity of CFGs

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

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

Variables: S, T, B

Productions:

\[
S \rightarrow T \mid B
\]

\[
T \rightarrow \text{bab} T1 \quad T \rightarrow \text{c} T2 \quad T \rightarrow \text{a} T3
\]

\[
B \rightarrow \text{cc} B1 \quad B \rightarrow \text{ab} B2 \quad B \rightarrow \text{ab} B3
\]

\[
T \rightarrow \text{bab1} \quad T \rightarrow \text{c}2 \quad T \rightarrow \text{a}3
\]

\[
B \rightarrow \text{cc1} \quad B \rightarrow \text{ab}2 \quad B \rightarrow \text{ab}3
\]
Ambiguity of CFGs

Each sequence of tiles gives a pair of derivations

\[ S \Rightarrow T \Rightarrow \text{bab} \quad T_1 \Rightarrow \text{babc} \quad T_2 \Rightarrow \text{babcc} \]

\[ S \Rightarrow B \Rightarrow \text{cc} \quad B_1 \Rightarrow \text{ccab} \quad B_2 \Rightarrow \text{ccabab} \]

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

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

\[
\begin{align*}
S & \quad | \\
T & \quad | \\
a_1 & \quad T \\
a_2 & \quad \ldots \quad n_1 \\
T & \quad | \\
a_i & \quad \quad n_i
\end{align*}
\]

\[
\begin{align*}
S & \quad | \\
B & \quad | \\
b_1 & \quad B \quad m_1 \\
b_2 & \quad \ldots \quad m_2 \\
B & \quad | \\
b_j & \quad m_j
\end{align*}
\]

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