Context-Free Languages

CS 712
Dept. of Computer Science
Univ. of New Hampshire
Context-Free Languages are generated by Context-Free Grammars

Context-Free Grammars have productions of the form: $A \rightarrow w$

- one nonterminal
- string of terminals and non-terminals
we know $\varepsilon a^\ast b^n / 4 \geq 3$ is not regular, but it is context-free.

\[
S \rightarrow \lambda \\
S \rightarrow aSb
\]

\[
S \rightarrow \lambda \\
S \rightarrow aSb \rightarrow \varepsilon b \\
S \rightarrow aSb \rightarrow \varepsilon SbS \rightarrow \varepsilon b \varepsilon b \varepsilon \\
S \rightarrow aSb \rightarrow \varepsilon SbS \rightarrow \varepsilon b \varepsilon b \varepsilon
\]
therefore the set of regular languages is a proper subset of the set of context-free languages.

remember: every regular language is context-free because it can be generated by a regular grammar and every regular grammar is context-free
Pushdown Automata (PDA)

A finite automaton with a stack

- The stack is initialized to contain a single symbol
- The stack has usual push/pop plus nop
- Arcs of the finite automaton are labeled

\[ \frac{L, S}{op} \quad i \xrightarrow{\frac{L, S}{op}} j \]

- If PDA is in state \( i \), and either \( L \) is next in the input or \( L \) is \( \lambda \), and the symbol on top of the stack is \( S \), then execute \( op \) and move to state \( j \).

If \( L \neq \lambda \), then go to next input.

Instruction written as a tuple

\( (i, L, S, op, j) \)
A string is accepted by the PDA if the stack is empty and the input string is consumed (i.e. no requirement that PDA be in any particular state)
PDA is deterministic if there is at most one move possible from each state.

There is nondeterminism if a state has

1) two arcs labelled with some input & stack symbols

2) two arcs labelled with some stack symbol and one arc is labelled with $\lambda$ for the input symbol
Theorem: The language generated by any context-free grammar can be accepted by a PDA.

Proof by construction:

Given CFG, construct a PDA with a single state:

i) Initialize stack to contain start symbol

ii) For each terminal symbol \( a \), create the instruction \((\phi, a, a, \text{pop}, \phi)\)

iii) For each production \( A \rightarrow B_1 B_2 \ldots B_n \), \( n \geq 0 \), create the instruction

\[
(\phi, \lambda, A, \text{pop}, \text{push } B_n, \text{push } B_{n-1}, \ldots, \text{push } B_1, \gamma, \phi)
\]

iv) For each production \( A \rightarrow \lambda \), create the instruction \((\phi, \lambda, A, \text{pop}, \phi)\)
The PDA accepts the language because each state transition with a nonterminal on top of the stack corresponds to one derivation step.
**Example**

\[ S \rightarrow \lambda \]
\[ S \rightarrow a \cdot S_b \]

<table>
<thead>
<tr>
<th>Input</th>
<th>Stack</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>999, bbb</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>999, bbb</td>
<td>S, b</td>
<td>9, S, b</td>
</tr>
<tr>
<td>99 bbb</td>
<td>S, b, bbb</td>
<td>9, 9, S, bbb</td>
</tr>
<tr>
<td>9 bb b</td>
<td>S, b, bbb, b</td>
<td>9, 9 bbb, b</td>
</tr>
<tr>
<td>b b b</td>
<td>S, b, bbb, bbb</td>
<td>9, 9, 9, bbb, b</td>
</tr>
<tr>
<td>b b b</td>
<td>S, b, bbb, bbb</td>
<td>9, 9, 9, bbb, b</td>
</tr>
<tr>
<td>b b</td>
<td>S, b, bbb, bbb</td>
<td>9, 9, 9, bbb, b</td>
</tr>
<tr>
<td>b</td>
<td>S, b, bbb, bbb</td>
<td>9, 9, 9, bbb, b</td>
</tr>
<tr>
<td>b</td>
<td>S, b, bbb, bbb</td>
<td>9, 9, 9, bbb, b</td>
</tr>
<tr>
<td>b</td>
<td>S, b, bbb, bbb</td>
<td>9, 9, 9, bbb, b</td>
</tr>
</tbody>
</table>

Diagram:

- \( S \rightarrow S_a \cdot S_b \)
deterministic PDA to match the same language

\[
\begin{array}{c|c|c}
\text{Input} & \text{Stack} \\
\hline
9a & 5 \\
a & s5b \\
b & sbb \\
b & b \\
\end{array}
\]

but in general, PDA relies on nonunimism to "choose" the correct production when a nonterminal is on top of the stack.

Also, parse is top-down with leftmost derivation, i.e., leftmost nonterminal in sentential form is rewritten in an intermediate step in a derivation.
non determinism is sometimes necessary

consider the even palindromes \( \text{\texttt{ww}w^R / \texttt{w} \in \{a,b\}^*} \)

deterministic PDA cannot tell when the middle of the string has been reached

\[
S \rightarrow \lambda \\
S \rightarrow aSa \\
S \rightarrow bSb
\]
LL(1) parsing

- leftmost derivation
- read input left to right

Top down
must always choose correct production
with 1 symbol lookahead
only parses a subset of the CFL's
can be implemented with a table

Select \([A, a] = \text{rhs of production with } A \text{ as lhs}\)