Recitation: Equivalence of PDAs Accepting by Final State / Empty Stack

11-711: Algorithms for NLP

October 24, 2014
Definitions

Reminder:

- **Empty stack:**

\[ L(M) = \{ w \in \Sigma^* | (q_0, w, Z) \vdash_M^* (q, \varepsilon, \varepsilon) \} \]

You must consume the entire input and empty the stack.

- **Final state:**

\[ L(N) = \{ w \in \Sigma^* | (q_0, w, Z) \vdash_N^* (f, \varepsilon, \gamma) \text{ s.t. } f \in F, \gamma \in \Gamma^* \} \]

You must consume the entire input and be in a final state. You may have leftovers in your stack.
An Example PDA Accepting by Empty Stack

Example PDA $N$ accepting by empty stack:

- $a, \gamma \rightarrow A\gamma$
- $b, A \rightarrow \epsilon$
- $c, \gamma \rightarrow \gamma$
- $\epsilon, Z \rightarrow \epsilon$

$\Sigma = \{a, b, c\}$
$\Gamma = \{Z, A\}$
An Example PDA Accepting by Empty Stack

Example PDA $N$ accepting by empty stack:

$$a, \gamma \rightarrow A\gamma$$  

$$b, A \rightarrow \epsilon$$

$$c, \gamma \rightarrow \gamma$$  

$$\epsilon, Z \rightarrow \epsilon$$

$\Sigma = \{a, b, c\}$

$\Gamma = \{Z, A\}$

What language does $N$ accept?
An Example PDA Accepting by Empty Stack

Example PDA $N$ accepting by empty stack:

$$
\begin{align*}
a, \gamma & \rightarrow A\gamma \\
\epsilon, Z & \rightarrow \epsilon \\
b, A & \rightarrow \epsilon \\
c, \gamma & \rightarrow \gamma
\end{align*}
$$

$\Sigma = \{a, b, c\}$
$\Gamma = \{Z, A\}$

What language does $N$ accept?
Balanced parentheses ($a = \“(\”, b = \“\)\”) ), with “c”s inserted anywhere.
Equivalent CFG

Equivalent to:

\[
G
\]

\[
S \rightarrow SS
\]

\[
S \rightarrow aSb
\]

\[
S \rightarrow c
\]

\[
S \rightarrow \epsilon
\]
Converting to Accept By Final State

Original \( N \) (accepting by empty stack):

\[
\begin{align*}
    a, \gamma & \rightarrow A\gamma \\
    b, A & \rightarrow \epsilon \\
    c, \gamma & \rightarrow \gamma \\
    \epsilon, Z & \rightarrow \epsilon
\end{align*}
\]

\[\Sigma = \{a, b, c\}\]
\[\Gamma = \{Z, A\}\]
Converting to Accept By Final State

Add a new stack symbol $\bot$, pronounced “bottom”:

$$a, \gamma \rightarrow A\gamma \quad b, A \rightarrow \epsilon$$

$$c, \gamma \rightarrow \gamma \quad \epsilon, Z \rightarrow \epsilon$$

$\Sigma' = \{a, b, c\}$
$\Gamma' = \{Z, A, \bot\}$
Converting to Accept By Final State

Push $\bot$ onto the bottom of the stack right away:

\[
\begin{align*}
    a, \gamma & \rightarrow A\gamma \\
    b, A & \rightarrow \epsilon \\
    c, \gamma & \rightarrow \gamma \\
    \epsilon, Z & \rightarrow Z\bot \\
    \epsilon, Z & \rightarrow \epsilon
\end{align*}
\]

$\Sigma' = \{a, b, c\}$

$\Gamma' = \{Z, A, \bot\}$
Move to a final state when your stack is exhausted:

\[
\begin{align*}
    a, \gamma & \rightarrow A\gamma \\
    b, A & \rightarrow \epsilon \\
    c, \gamma & \rightarrow \gamma \\
    \epsilon, Z & \rightarrow Z\perp \\
    \epsilon, \perp & \rightarrow \epsilon \\
    \epsilon, Z & \rightarrow \epsilon
\end{align*}
\]

\[\Sigma' = \{a, b, c\}\]
\[\Gamma' = \{Z, A, \perp\} \]
Done! Note that there are no edges out of $f$. If you move to $f$ before you’ve read the entire input, that computation path will get stuck.

$$\Sigma' = \{a, b, c\}$$
$$\Gamma' = \{Z, A, \bot\}$$
Converting to Accept By Final State

Why do we need \( \perp \)?

Because you need a transition that can check whether the stack is actually exhausted. If the stack is actually exhausted, you're already stuck, and can't accept!
Converting to Accept By Final State

Why do we need \( \bot \)?

Because you need a transition that can check whether the stack is exhausted or not.
Why do we need ⊥?

Because you need a transition that can check whether the stack is exhausted or not.

If the stack is *actually* exhausted... you’re already stuck, and can’t accept!
An Example PDA Accepting by Final State

Example PDA $M$ accepting by final state:

\[
b, A \rightarrow \epsilon
\]

\[
b, A \rightarrow \epsilon
\]

\[
\Sigma = \{a, b\}
\]

\[
\Gamma = \{Z, A\}
\]
An Example PDA Accepting by Final State

Example PDA $M$ accepting by final state:

$\Sigma = \{a, b\}$
$\Gamma = \{Z, A\}$

What language does $M$ accept?
Example PDA $M$ accepting by final state:

$$b, A \rightarrow \epsilon$$

$$a, \gamma \rightarrow A\gamma$$

$$\gamma \rightarrow A\gamma$$

$$b, A \rightarrow \epsilon$$

$\Sigma = \{a, b\}$

$\Gamma = \{Z, A\}$

What language does $M$ accept?

“Implicitly closed” parens with an odd number of open parens. You may leave parens unclosed, but you may never close them before you open them.
Equivalent to:

\[ G \]

\[ S \rightarrow O \]
\[ O \rightarrow EO \]
\[ O \rightarrow OE \]
\[ O \rightarrow aEB \]
\[ E \rightarrow OO \]
\[ E \rightarrow EE \]
\[ E \rightarrow aOB \]
\[ E \rightarrow \epsilon \]
\[ B \rightarrow b \]
\[ B \rightarrow \epsilon \]

(Can probably be made smaller?)
Converting to Accept By Empty Stack

Original $M$ (accepting by final state):

- $b, A \rightarrow \epsilon$
- $a, \gamma \rightarrow A\gamma$
- $a, \gamma \rightarrow A\gamma$

$\Sigma = \{a, b\}$
$\Gamma = \{Z, A\}$
Converting to Accept By Empty Stack

Add a *drain* state, with a transition from every final state:

\[ b, A \rightarrow \epsilon \]
\[ a, \gamma \rightarrow A\gamma \]
\[ \epsilon, \gamma \rightarrow \epsilon \]

\[ \Sigma = \{a, b\} \]
\[ \Gamma = \{Z, A\} \]
Converting to Accept By Empty Stack

Done!
Note that once you move to *drain*, you never read any more input. If you haven’t already consumed the entire input, you are stuck.

\[ b, A \rightarrow \epsilon \]

\[ a, \gamma \rightarrow A\gamma \]

\[ \epsilon, \gamma \rightarrow \epsilon \]

\[ \Sigma = \{a, b\} \]

\[ \Gamma = \{Z, A\} \]
Why do we need \textit{drain}? (Why not just stay at \textit{f} and drain?)
Converting to Accept By Empty Stack

Why do we need *drain*? (Why not just stay at $f$ and drain?)

Because then you could drain a little, then continue the computation.

We only want to accept when the input has been entirely consumed.
Recitation: Equivalence of PDAs Accepting by Final State / Empty Stack

11-711: Algorithms for NLP

October 24, 2014