CS 172: Computability and Complexity

Equivalence of CFGs and PDAs & CFL Pumping Lemma

Sanjit A. Seshia
EECS, UC Berkeley

Acknowledgments: L.von Ahn, L. Blum, M. Blum
Formal Definition of **Acceptance**

PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, F) \) accepts a word \( w \in \Sigma^* \) where \( w = w_1w_2w_3 \ldots w_m \) with \( w_i \in \Sigma_\epsilon \) if there exists a sequence

\[
(q_0, s_0) \rightarrow (q_1, s_1) \rightarrow (q_2, s_2) \rightarrow \ldots \rightarrow (q_m, s_m)
\]

where

- \( s_i \in \Gamma^* \) (represent the stack), with \( s_0 = \epsilon \),
- \( q_m \in F \),
- \((q_{i+1}, b) \in \delta(q_i, w_{i+1}, a)\)

where \( s_i = at \) and \( s_{i+1} = bt \), \( a, b \in \Gamma_\epsilon \), \( t \in \Gamma^* \)
Theorem

Suppose \( L \) is generated by a CFG \( G = (V, \Sigma, R, S) \)

Construct \( P = (Q, \Sigma, \Gamma, \delta, q, F) \) that recognizes \( L \)

A Language is generated by a CFG

\[ \iff \]

It is recognized by a PDA
A Language is generated by a CFG
⇒
It is recognized by a PDA

\[ \varepsilon, \varepsilon \rightarrow S\$ \]

\[ q_0 \]

\[ \varepsilon, A \rightarrow w \text{ for rule } A \rightarrow w \]

\[ a,a \rightarrow \varepsilon \text{ for terminal } a \]

\[ \varepsilon, \$ \rightarrow \varepsilon \]

\[ q_{\text{loop}} \]

\[ \varepsilon, \varepsilon \rightarrow S\$ \]

\[ q_f \]
S \rightarrow aTb
T \rightarrow Ta \mid \varepsilon
Theorem

Suppose PDA $P = (Q, \Sigma, \Gamma, \delta, q, F)$ recognizes $L$
Construct CFG $G = (V, \Sigma, R, S)$ that generates $L$

A Language is generated by a CFG
$\iff$
It is recognized by a PDA
Proof Ideas

- \( A_{pq} = \) variable generating all \( x \) that takes \( P \) from \((p, \varepsilon)\) to \((q, \varepsilon)\)

- Formal construction:
  - \( V = \{ A_{pq} \mid p, q \in Q \} \)
  - \( S = A_{q0qf} \)
  - Defining \( R \):
    
    Intuition: Derivations correspond to computations of \( P \);
    There are two cases for a derivation from \( A_{pq} \)
    1. Stack is empty only at the beginning and end of the derivation from \( A_{pq} \)
    2. Stack becomes empty somewhere in between
Case 1

\[ A_{pq} \rightarrow aA_{rs}b \]

Case 2

\[ A_{pq} \rightarrow A_{pr}A_{rq} \]
Recap of Proof Ideas

- $A_{pq} =$ variable generating all $x$ that takes $P$ from $(p, \varepsilon)$ to $(q, \varepsilon)$

- Formal construction:
  - $V = \{ A_{pq} \mid p, q \in Q \}$
  - $S = A_{q0qf}$
  - $R$ defined as follows:
    - $A_{pp} \rightarrow \varepsilon \quad \forall p \in Q$
    - $A_{pq} \rightarrow a A_{rs} b \quad \forall p, q, r, s \in Q$
    - $s.t. (r, t) \in \delta(p, a, \varepsilon)$
    - $(q, \varepsilon) \in \delta(s, b, t)$
    - $A_{pq} \rightarrow A_{pr} A_{rq} \quad \forall p, q, r \in Q$

[Proof sketched on whiteboard, see textbook for details]
CFL Pumping Lemma

Let $L$ be a context-free language

Then there exists $p$ such that for all $w \in L$ and $|w| \geq p$

we can write $w = uvxyz$, where:

1. $uv^i xy^i z \in L$ for any $i \geq 0$
2. $|vy| > 0$
3. $|vxy| \leq p$
“Surgery” on Parse Trees

Idea: If w is long enough, then any parse tree for w must have a path that contains a variable more than once