

Lec 10. Pushdown Automata and CFG

Eunjung Kim

PDA FOR $L = \{w \cdot w^R : w \in \{0, 1\}^*\}$

PDA FOR $L = \{w \in \{a, b\}^* : |w|_a = |w|_b\}$

EQUIVALENCE OF CFG AND PDA

THEOREM

A language is context-free if and only if some pushdown automaton recognizes it.

- (\Rightarrow) : converting a CFG to a PDA.
- (\Leftarrow) : converting a PDA to a CFG.

CONVERTING CFG TO PDA

- From a context-free grammar $G = (V, \Sigma, R, S)$, we aim to construct a PDA P such that $L(P) = L(G)$.

CONVERTING CFG TO PDA

- From a context-free grammar $G = (V, \Sigma, R, S)$, we aim to construct a PDA P such that $L(P) = L(G)$.
- Key idea: we design PDA P which **simulates a leftmost derivation of w** .

CONVERTING CFG TO PDA

- From a context-free grammar $G = (V, \Sigma, R, S)$, we aim to construct a PDA P such that $L(P) = L(G)$.
- Key idea: we design PDA P which **simulates a leftmost derivation of w** .
 - 1 It "matches" the input symbol and the stack symbol if the stack symbol is an element of Σ .
 - 2 It "replaces" the stack symbol A by γ if A is a variable of G and there is a rule $A \rightarrow \gamma$.

CONVERTING CFG TO PDA: EXAMPLE

$L = \{0^n 1^n : n \geq 0\}$ is the language of the grammar $S \rightarrow 0S1 \mid \epsilon$.

CONVERTING CFG TO PDA

Construct a PDA P as follows.

- 1 There are three states q_{start}, q, q_{accept} .
- 2 The stack alphabet is $V \cup \Sigma \cup \{\$, \}\}$.
- 3 Initially, P places the marker $\$$ onto the (empty) stack, then the start symbol S on the CFG G .
- 4 It loops at the state q and executes the following unless the stack symbol is $\$$
 - If the stack symbol is $A \in V$, then P nondeterministically chooses a rule of the form $A \rightarrow \gamma$ and pushes γ onto stack so that the first symbol of γ is at the top.
 - If the stack symbol is $a \in \Sigma$, then P reads the symbol $a \in \Sigma$ in the input and stays in the current state. If a cannot be read ("does not match"), no move is defined and the current computation branch dies out.
- 5 If the stack symbol is $\$$, then it goes to q_{accept} . The input string is accepted if the string has been read fully. If not, the current branch dies out.

CONVERTING CFG TO PDA

Construct a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ from $G = (V, \Sigma, R, S)$:

- 1 $Q = \{q_{start}, q, q_{accept}\}$. $\Gamma = V \cup \Sigma \cup \{\$\}$.
- 2 $\delta(q_{start}, \epsilon, \epsilon) = \{(q, S\$)\}$.
- 3 For each stack symbol in $V \cup \Sigma \cup \{\$\}$
 - for every $A \in V$: $\delta(q, \epsilon, A) = \{(q, \gamma) : \text{for all rules } A \rightarrow \gamma \text{ in } G\}$
 - for every $a \in \Sigma$: $\delta(q, a, a) = \{(q, \epsilon)\}$
 - $\delta(q, \epsilon, \$) = \{q_{accept}, \epsilon\}$.

CONVERTING CFG TO PDA

Construct a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ from $G = (V, \Sigma, R, S)$:

- 1 $Q = \{q_{start}, q, q_{accept}\}$. $\Gamma = V \cup \Sigma \cup \{\$\}$.
- 2 $\delta(q_{start}, \epsilon, \epsilon) = \{(q, S\$)\}$.
- 3 For each stack symbol in $V \cup \Sigma \cup \{\$\}$
 - for every $A \in V$: $\delta(q, \epsilon, A) = \{(q, \gamma) : \text{for all rules } A \rightarrow \gamma \text{ in } G\}$
 - for every $a \in \Sigma$: $\delta(q, a, a) = \{(q, \epsilon)\}$
 - $\delta(q, \epsilon, \$) = \{q_{accept}, \epsilon\}$.

How to implement a transition such as $\{(q, \gamma) \in \delta(q, \epsilon, A)$ when γ is a string, not necessarily a symbol in Γ_ϵ ?

CONVERTING PDA TO CFG

Step A. Streamlining the PDA.

- 1 It has a single accept state q_{accept} .
- 2 It empties its stack before accepting.
- 3 Each transition move either pushes a symbol onto the stack (push move) or pops a symbol off the stack (pop move), but does not do both at the same time.

CONVERTING PDA TO CFG

Step B. Variables A_{pq} for all $p, q \in Q$.

- 1 Meaning of A_{pq} : we intend to design CFG G so that

$$L(A_{pq}) := \{w : A \Rightarrow_G^* w\}$$

coincides with

$$\{w : (p, w, \epsilon) \vdash_G^* (q, \epsilon, \epsilon)\}$$

- 2 Take A_{st} as the start variable of CFG G , where $s = q_0$ and $t = q_{accept}$.
3 Then $L(G)(= L(A_{st}))$ coincides with

$$\{w : (q_0, w, \epsilon) \vdash_G^* (q_{accept}, \epsilon, \epsilon)\},$$

which is precisely $L(P)$.

CONVERTING PDA TO CFG

Step C. Designing a production rule for the variable A_{pq} .

1 For a string w in

$$\{w : (p, w, \epsilon) \vdash_G^* (q, \epsilon, \epsilon)\},$$

two situation can occur when P runs on w .

- A the stack gets empty while running
- B the symbol pushed at the beginning is never popped till the last moment.

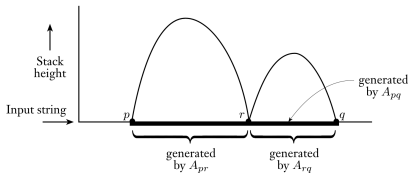


Figure 2.28, Sipser 2012

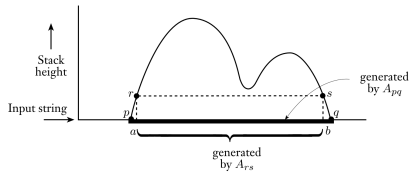


Figure 2.29, Sipser 2012

CONVERTING PDA TO CFG

Step C. Designing a production rule for the variable A_{pq} .

1 For a string w in

$$\{w : (p, w, \epsilon) \vdash_G^* (q, \epsilon, \epsilon)\},$$

two situation can occur when P runs on w .

A the stack gets empty while running:

i.e. $w \in L(A_{pr}) \cdot L(A_{rq})$.

B the symbol pushed at the beginning is never popped till the last moment.

i.e. $w \in aL(rs)b$ for all $a, b \in \Sigma$ whenever $\delta(p, a, \epsilon)$ contains (r, u) and $\delta(s, b, u)$ contains (q, ϵ) for some $u \in \Gamma$.

Each case is simulated by the next rules.

- case A: $A_{pq} \rightarrow A_{pr}A_{rq}$ for all $p, q, r \in Q$
- case B: $A_{pq} \rightarrow aA_{rs}b$ for all $p, q, r, s \in Q$ and $a, b \in \Sigma_\epsilon$, and $u \in \Gamma$ such that $\delta(p, a, \epsilon)$ contains (r, u) and $\delta(s, b, u)$ contains (q, ϵ) .

The new CFG G contains all the above rules.

CONVERTING PDA TO CFG

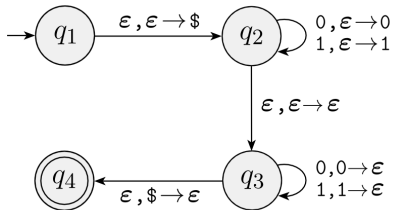


Figure 2.19, Sipser 2012