Equivalence of PDA, CFG

Conversion of CFG to PDA

Conversion of PDA to CFG

Overview

- CFG’s and PDA’s are both useful to deal with properties of the CFL’s like RE and DFA for regular languages.
- CFL’s can be described by CFG and CLF’s are processed by PDA’s, so CFG and PDA are equivalent.
- Any CFG can be converted into equivalent PDA representation and vice-versa

Converting a CFG to a PDA

- Let L = L(G).
- Construct PDA P such that L(P) = L.
- P has:
  - One state q.
  - Input symbols = terminals of G.
  - Stack symbols = all symbols of G.
  - Start symbol = start symbol of G.
- Here P will accept L by empty stack.

Intuition About P

- Given input w, P will step through a leftmost derivation of w from the start symbol S.
- Since P can't know what this derivation is, or even what the end of w is, it uses nondeterminism to "guess" the production to use at each step.

Intuition -(2)

- At each step, P represents some *left-sentential form* (step of a leftmost derivation).
- If the stack of P is α, and P has so far consumed x from its input, then P represents left-sentential form $x\alpha$.
- At empty stack, the input consumed is a string in L(G).

Transition Function of P

1. $\delta(q, a, a) = (q, \epsilon)$ for all $a \in T$ (*Type 1* rules)
   - This step does not change the Left Sentencial Form represented, but "moves" responsibility for *a* from the stack to the consumed input.
2. If $A \to \alpha$ is a production of G, then $\delta(q, \epsilon, A)$ contains (q, α). (*Type 2* rules)
   - Guess a production for A, and represent the next LSF in the derivation.

Example: CFG to PDA

- **G=({E,T,F}, {(,),a,+,*}, P, E)** where P is :
  - E→ T|E+T
  - T→ T*F | F
  - F→ a | (E)
- We define P(G) = { $\{q_0\}$,{(,),a,+,*} ,{ E,T,F,(,),a,+,*}, δ , q , E) that accepts L(G) by empty stack.
- δ is defined as
  1. **δ(q_0, ϵ , E) = { (q_0, T) , (q_0, E+T) } by rule 2**
  2. **δ(q_0, ϵ , T) = { (q_0, T*F) , (q_0, F) } by rule 2**
  3. **δ(q_0, ϵ , F) = { (q_0, a) , (q_0, (E)) } by rule 2**
  4. **δ(q_0, (, ( ) = { (q_0, ϵ) by rule 1**
  5. **δ(q_0,),)) = { (q_0, ϵ) by rule 1**
  6. **δ(q_0, a, a) = { (q_0, ϵ) by rule 1**
  7. **δ(q_0, +, +) = { (q_0, ϵ) by rule 1**
  8. **δ(q_0, * , *) = { (q_0, ϵ) by rule 1**

Acceptance of string by P

- Let string input w= (a*a)
- The initial ID for P is: (q_0,(a*a),E)
- So processing of string by P is given by

(q_0,(a*a),E) ⊦ (q_0, (a*a), T)

⊦ (q_0, (a*a), F)

⊦ (q_0, (a*a), (E))

⊦ (q_0, a*a), E))

⊦ (q_0, a*a), T))

⊦ (q_0, a*a), T*F))

⊦ (q_0, a*a), a * F))

⊦ (q_0, *a), *F))

⊦ (q_0, a), F))

⊦ (q_0, a), a))

⊦ (q_0,),))

⊦ (q_0, ϵ, ϵ) Accept.

Exercise

- Convert following Grammar in to PDA
  - $S \rightarrow aAA$
  - $A \rightarrow aS \mid bS \mid a$

From a PDA to a CFG

- Now, assume L = L(P).
- We'll construct a CFG G such that L = L(G).
- Intuition: G will have variables generating exactly the inputs that cause P to have the net effect of popping a stack symbol X while going from state p to state q.
  - P never gets below this X while doing so.

PDA to CFG

- Given a PDA $P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0)$ accepting language by empty stack, we can define equivalent CFG as
- G =(V,T,P,S_ where
  - $V = \{S\} \cup \{[pXq] \mid X \in \Gamma \text{ and } p,q \in Q\}$
  - $T = \Sigma$
  - S = Start symbol of G.
  - P contains the following (next slide)

PDA to CFG

1. For every $q \in Q$, the production $S \rightarrow [qZ_0q]$ is in P
2. For every $q, q_1 \in Q$, $a \in \Sigma \cup \{\epsilon\}$ and $X \in \Gamma$ is $\delta(q,a,X)$ contains (q_1, ϵ) the $[qXq_1] \rightarrow a$ is in P
3. For every $q, q_1 \in Q$, $a \in \Sigma \cup \{\epsilon\}$ and $X \in \Gamma$ and $m>=1$
4. $\delta(q,a,X) = (q_1,Y_1Y_2Y_3 \ldots \ldots Y_m)$ for some $Y_1, Y_2, Y_3, \ldots \ldots, Y_m) \in \Gamma$, then for every choice of $q_2, q_3, \ldots q_{m+1}$ in Q the production

$$[qXq_{m+1}] \rightarrow [q_1Y_1q_2][q_2Y_2q_3][q_3Y_3q_4] \ldots [q_mY_mq_{m+1}] \text{ is in P}$$

Variables of G

- G’s variables are of the form [pXq].
- This variable generates all and only the strings w such that
  $(p, w, X) \vdash^* (q, \epsilon, \epsilon)$.
- Also a start symbol S we’ll talk about later.

Productions of G

- Each production for [pXq] comes from a move of P in state p with stack symbol X.
- Simplest case: **δ(p, a, X) contains (q, ϵ)**.
- Then the production is **[pXq] -> a.**
  - Note *a* can be an input symbol or ϵ.
- Here, [pXq] generates *a*, because reading *a* is one way to pop X and go from p to q.

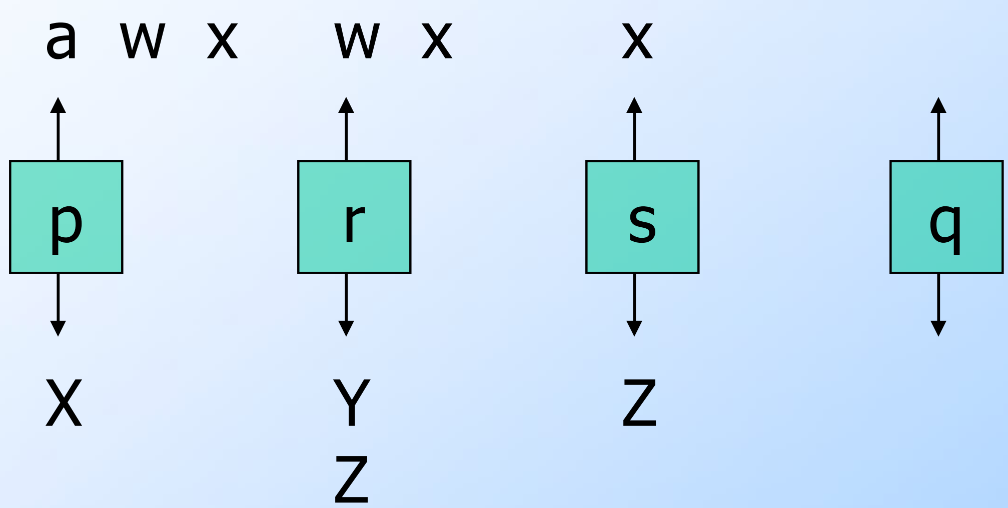
Productions of G – (2)

- Next simplest case: $\delta(p, a, X)$ contains (r, Y) for some state r and symbol Y.
- G has production [pXq] -> a[rYq].
  - We can erase X and go from p to q by reading *a* (entering state r and replacing the X by Y) and then reading some w that gets P from r to q while erasing the Y.
- Note: [pXq] =>* aw whenever [rYq] =>* w.

Productions of G – (3)

- Third simplest case: $\delta(p, a, X)$ contains (r, YZ) for some state r and symbols Y and Z.
- Now, P has replaced X by YZ.
- To have the net effect of erasing X, P must erase Y, going from state r to some state s, and then erase Z, going from s to q.

Picture of Action of P



Third-Simplest Case – Concluded

- Since we do not know state s, we must generate a family of productions:

  [pXq] -> a[rYs][sZq]

  for all states s.
- [pXq] =>* awx whenever [rYs] =>* w and [sZq] =>* x.

Productions of G: General Case

- Suppose $\delta(p, a, X)$ contains $(r, Y_1, \ldots Y_k)$ for some state r and $k \geq 3$.
- Generate family of productions

[pXq] ->

$$a[rY_1s_1][s_1Y_2s_2]\ldots[s_{k-2}Y_{k-1}s_{k-1}][s_{k-1}Y_kq]$$

Completion of the Construction

- We can prove that $(q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)$ if and only if $[q_0 Z_0 p] \Rightarrow^* w$.
  - Proof is in text; it is two easy inductions.
- But state p can be anything.
- Thus, add to G another variable S, the start symbol, and add productions $S \rightarrow [q_0 Z_0 p]$ for each state p.