CFL → PDA → ε-NFA + Stack

↘ CFG

Equivalence of PDA, CFG

Conversion of CFG to PDA

Conversion of PDA to CFG

Overview

- When we talked about closure properties of regular languages, it was useful to be able to jump between RE and DFA representations.
- Similarly, CFG's and PDA's are both useful to deal with properties of the CFL's.

Overview – (2)

- Also, PDA's, being "algorithmic," are often easier to use when arguing that a language is a CFL. Context-free language
- Example: It is easy to see how a PDA can recognize balanced parentheses; not so easy as a grammar.
- But all depends on knowing that CFG's and PDA's both define the CFL's.

Converting a CFG to a PDA

Grammers.

- Let L = L(G). CFG
- Construct PDA P such that N(P) = L. PDA
- P has:
  - One state q.
  - Input symbols = terminals of G.
  - Stack symbols = all symbols of G.
  - Start symbol = start symbol of G.

PDA CFG

Intuition About P

string

- 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)

Derivate
Start
Symbol

- 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. $\delta(q, a, a) = (q, \epsilon)$. (*Type 1* rules)
   - This step does not change the LSF represented, but "moves" responsibility for *a* from the stack to the consumed input.
2. If $A \rightarrow \alpha$ 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.

Proof That L(P) = L(G)

- We need to show that $(q, wx, S) \vdash^* (q, x, \alpha)$ for any x if and only if $S =>^*_{lm} w\alpha$.
- Part 1: "only if" is an induction on the number of steps made by P.
- Basis: 0 steps.
  - Then $\alpha = S$, $w = \epsilon$, and $S =>^*_{lm} S$ is surely true.

From a PDA to a CFG

- Now, assume L = N(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.

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.

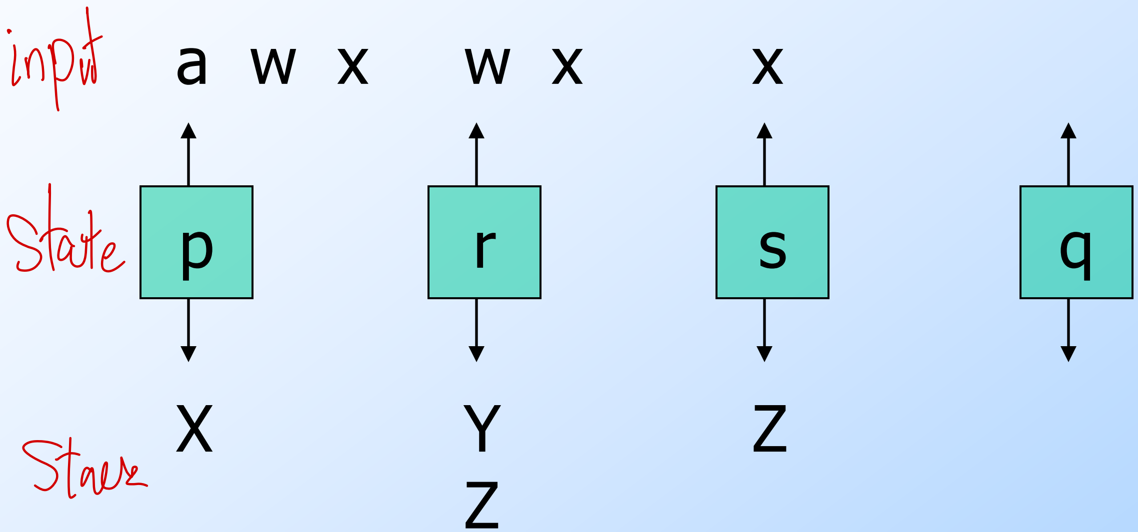
Productions of G – (2)

- 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:

  top of stack

  [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

  $[pXq] \rightarrow$

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