Unit 3

Left Recursion

A production of grammar is said to have **left recursion** if the leftmost variable of its RHS is same as variable of its LHS.

- A grammar containing a production having left recursion is called as Left Recursive Grammar.

- **Example-**
- $S \rightarrow Sa / \epsilon$
- (**Left Recursive Grammar**)

Elimination of Left Recursion

- Left recursion is eliminated by converting the grammar into a right recursive grammar.
- If we have the left-recursive pair of productions-
- **A → Aα / β**
- (Left Recursive Grammar)
- where β does not begin with an A.

- Then, we can eliminate left recursion by replacing the pair of productions with-
- **$A \rightarrow \beta A'$**
- **$A' \rightarrow \alpha A' / \in$**
- (Right Recursive Grammar)
- This right recursive grammar functions same as left recursive grammar.

Greibach Normal Form (GNF)

- A CFG $G = (V, T, R, S)$ is said to be in GNF if every production is of the form $A \to a\alpha$, where $a \in T$ and $\alpha \in V*$, i.e., α is a string of zero or more variables.
- Definition: A production $U \in R$ is said to be in the form left recursion, if $U : A \to A\alpha$ for some $A \in V$

The stepwise algorithm is as follows:

- 1. Eliminate null productions, unit productions and useless symbols from the grammar G and then construct a G0 = (V 0 , T, R0 , S) in Chomsky Normal Form (CNF) generating the language L(G0) = L(G) − {∈}.
- 2. Rename the variables like A1, A2, . . . An starting with S = A1.
- 3. Modify the rules in R0 so that if $Ai \to Aj\gamma \in R0$ then $j > i$

- 4. Starting with A1 and proceeding to An this is done as follows:

  (a) Assume that productions have been modified so that for $1 \leq i \leq k$, $Ai \to Aj\gamma \in R0$ only if $j > i$

  (b) If $Ak \to Aj\gamma$ is a production with $j < k$, generate a new set of productions substituting for the Aj the body of each Aj production.

  (c) Repeating (b) at most $k - 1$ times we obtain rules of the form $Ak \to Ap\gamma$, $p \geq k$

  (d) Replace rules $Ak \to Ak\gamma$ by removing left-recursion as stated above.

- 5. Modify the $Ai \to Aj\gamma$ for $i = n-1, n-2, ., 1$ in desired form at the same time change the Z production rules.

- Example: Convert the following grammar G into Greibach Normal Form (GNF).

$S \rightarrow XA \mid BB$

$B \rightarrow b \mid SB$ $X \rightarrow b$

$A \rightarrow a$

To write the above grammar G into GNF, we shall follow the following steps:

1. Rewrite G in Chomsky Normal Form (CNF) It is already in CNF.
2. Re-label the variables

   S with A1

   X with A2

   A with A3

   B with A4

- 3. Identify all productions which do not conform to any of the types listed below:

  $Ai \rightarrow A_j x_k$ such that $j > i$

  $Zi \rightarrow A_j x_k$ such that $j \leq n$

  $Ai \rightarrow ax_k$ such that $x_k \in V*$ and $a \in T$

- 4. A4 → A1A4 .... Identified

- 5. $A4 \rightarrow A1A4 \mid b$.

  To eliminate A1 we will use the substitution rule

- $A1 \rightarrow A2A3 \mid A4A4$.
- Therefore, we have
- $A4 \rightarrow A2A3A4 \mid A4A4A4 \mid b$
- The above two productions still do not conform to any of the types in step 3.
- Substituting for $A2 \rightarrow b$

$A4 \rightarrow bA3A4 \mid A4A4A4 \mid b$

Now we have to remove left recursive production

$A4 \rightarrow A4A4A4$

$A4 \rightarrow bA3A4 \mid b \mid bA3A4Z \mid bZ$

$Z \rightarrow A4A4 \mid A4A4Z$

6. All productions for A2, A3 and A4 are in GNF

- for A1 → A2A3 | A4A4
- Substitute for A2 and A4 to convert it to GNF
- A1 → bA3 | bA3A4A4 | bA4 | bA3A4ZA4 | bZA4
- for Z → A4A4 | A4A4Z
- Substitute for A4 to convert it to GNF
- Z →

  A3A4A4 | bA4 | bA3A4ZA4 | bZA4 | bA3A4A4Z | bA4Z | b

  A3A4ZA4Z | bZA4Z

7. Finally the grammar in GNF is

A1 → bA3 | bA3A4A4 | bA4 | bA3A4ZA4 | bZA4

A4 → bA3A4 | b | bA3A4Z | bZ

Z →

bA3A4A4 | bA4 | bA3A4ZA4 | bZA4 | bA3A4A4Z | bA4Z |

bA3A4ZA4Z | bZA4Z

A2 → b

A3 → a

Pushdown Automata

- The PDA is an automaton equivalent to the CFG in language-defining power.
- Only the nondeterministic PDA defines all the CFL's.
- But the deterministic version models parsers.
  - Most programming languages have deterministic PDA's.

INTUITION: PDA

- Think of an ε-NFA with the additional power that it can manipulate a stack.
- Its moves are determined by:
  1. The current state (of its "NFA"),
  2. The current input symbol (or ε), and
  3. The current symbol on top of its stack.

Intuition: PDA – (2)

- Being nondeterministic, the PDA can have a choice of next moves.
- In each choice, the PDA can:
  1. Change state, and also
  2. Replace the top symbol on the stack by a sequence of zero or more symbols.
     - Zero symbols = "pop."
     - Many symbols = sequence of "pushes."

PDA Formalism

- A PDA is described by:

1. A finite set of *states* (Q, typically).
2. An *input alphabet* (Σ, typically).
3. A *stack alphabet* (Γ, typically).
4. A *transition function* (δ, typically).
5. A *start state* (q_0, in Q, typically).
6. A *start symbol* (Z_0, in Γ, typically).
7. A set of *final states* ($F \subseteq Q$, typically).

Conventions

- a, b, … are input symbols.
  - But sometimes we allow **ε** as a possible value.
- …, X, Y, Z are stack symbols.
- …, w, x, y, z are strings of input symbols.
- α, β,… are strings of stack symbols.

The Transition Function

- Takes three arguments:
  1. A state, in Q.
  2. An input, which is either a symbol in Σ or ε.
  3. A stack symbol in Γ.
- δ(q, a, Z) is a set of zero or more actions of the form (p, α).
  - p is a state; α is a string of stack symbols.

Actions of the PDA

- If δ(q, a, Z) contains (p, α) among its actions, then one thing the PDA can do in state q, with a at the front of the input, and Z on top of the stack is:

1. Change the state to p.
2. Remove a from the front of the input (but a may be ε).
3. Replace Z on the top of the stack by α.

Example: PDA

- Design a PDA to accept $\{0^n1^n \mid n \geq 1\}$.
- The states:
  - q = start state. We are in state q if we have seen only 0's so far.
  - p = we've seen at least one 1 and may now proceed only if the inputs are 1's.
  - f = final state; accept.

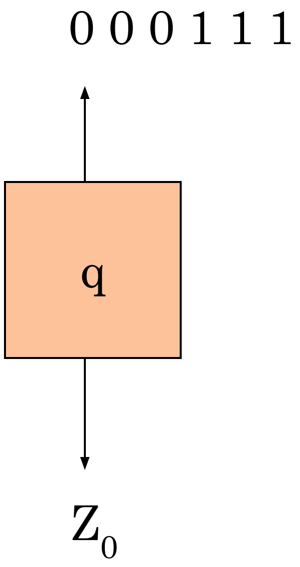
Example: PDA – (2)

- The stack symbols:
  - Z_0 = start symbol. Also marks the bottom of the stack, so we know when we have counted the same number of 1's as 0's.
  - X = marker, used to count the number of 0's seen on the input.

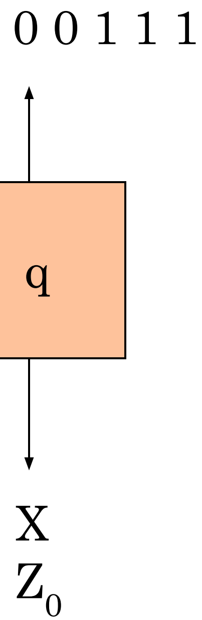
EXAMPLE: PDA – (3)

- The transitions:
  - $\delta(q, 0, Z_0) = \{(q, XZ_0)\}$.
  - $\delta(q, 0, X) = \{(q, XX)\}$. These two rules cause one X to be pushed onto the stack for each 0 read from the input.
  - $\delta(q, 1, X) = \{(p, \varepsilon)\}$. When we see a 1, go to state p and pop one X.
  - $\delta(p, 1, X) = \{(p, \varepsilon)\}$. Pop one X per 1.
  - $\delta(p, \varepsilon, Z_0) = \{(f, Z_0)\}$. Accept at bottom.

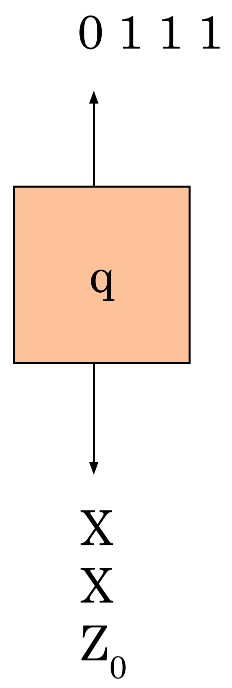
Actions of the Example PDA



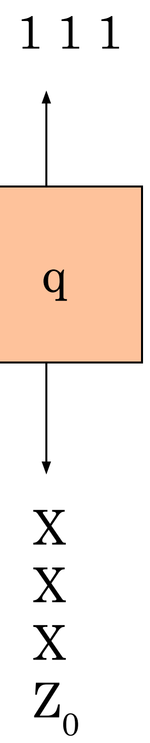
Actions of the Example PDA



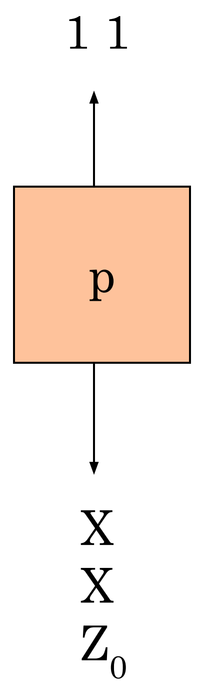
Actions of the Example PDA



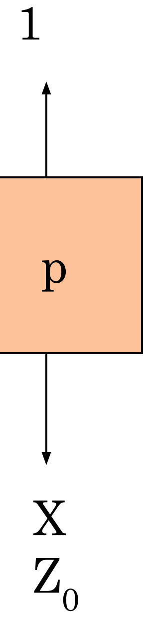
Actions of the Example PDA



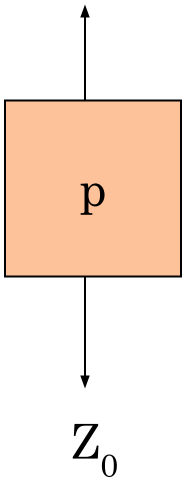
Actions of the Example PDA



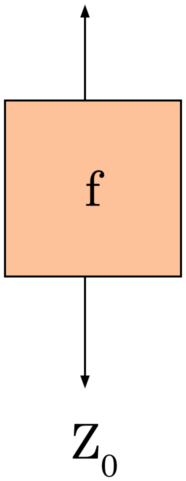
Actions of the Example PDA



Actions of the Example PDA



Actions of the Example PDA



Instantaneous Descriptions

- We can formalize the pictures just seen with an *instantaneous description* (ID).
- A ID is a triple (q, w, α), where:
  1. q is the current state.
  2. w is the remaining input.
  3. α is the stack contents, top at the left.

The "Goes-To" Relation

- To say that ID I can become ID J in one move of the PDA, we write $I \vdash J$.
- Formally, $(q, aw, X\alpha) \vdash (p, w, \beta\alpha)$ for any w and α, if $\delta(q, a, X)$ contains (p, β).
- Extend $\vdash$ to $\vdash^*$, meaning "zero or more moves," by:
  - Basis: $I \vdash^* I$.
  - Induction: If $I \vdash^* J$ and $J \vdash K$, then $I \vdash^* K$.

Example: Goes-To

- Using the previous example PDA, we can describe the sequence of moves by: $(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash (q, 0111, XXZ_0) \vdash (q, 111, XXXZ_0) \vdash (p, 11, XXZ_0) \vdash (p, 1, XZ_0) \vdash (p, \varepsilon, Z_0) \vdash (f, \varepsilon, Z_0)$
- Thus, $(q, 000111, Z_0) \vdash^* (f, \varepsilon, Z_0)$.
- What would happen on input 0001111?

Legal because a PDA can use ε input even if input remains.

Answer

- (q, 0001111, Z_0) ⊦ (q, 001111, XZ_0) ⊦ (q, 01111, XXZ_0) ⊦ (q, 1111, $XXXZ_0$) ⊦ (p, 111, XXZ_0) ⊦ (p, 11, XZ_0) ⊦ (p, 1, Z_0) ⊦ (f, 1, Z_0)
- Note the last ID has no move.
- 0001111 is not accepted, because the input is not completely consumed.

Language of a PDA

- The common way to define the language of a PDA is by *final state*.
- If P is a PDA, then L(P) is the set of strings w such that $(q_0, w, Z_0) \vdash^* (f, \varepsilon, \alpha)$ for final state f and any α.

Language of a PDA – (2)

- Another language defined by the same PDA is by *empty stack*.
- If P is a PDA, then N(P) is the set of strings w such that $(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$ for any state q.

Deterministic PDA's

- To be deterministic, there must be at most one choice of move for any state q, input symbol *a*, and stack symbol X.
- In addition, there must not be a choice between using input ε or real input.
- Formally, δ(q, a, X) and δ(q, ε, X) cannot both be nonempty.

END