Pushdown Automata

5.1 Introduction to Push Down Automata (PDA), Representation of PDA, Operations of PDA, Move of a PDA, Instantaneous Description for PDA

5.2 Deterministic PDA, Non Deterministic PDA, Acceptance of strings by PDA, Language of PDA

5.3 Construction of PDA by Final State , Construction of PDA by Empty Stack,

5.4 Conversion of PDA by Final State to PDA accepting by Empty Stack and vice-versa, Conversion of CFG to PDA, Conversion of PDA to CFG

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.

Introduction: PDA

- The PDA is a Abstract Machine which is considered to have input tape, finite control and a Stack.
- Its moves are determined by:
  1. The current state.
  2. The current input symbol (or ϵ), and
  3. The current symbol on top of its stack.

Introduction: PDA

- Being a nondeterministic, the PDA can have a choice of next moves.
- At each step, PDA can have either push or pop operation in its stack.
- 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 – Formal Definition

- A PDA is described by 7 tuples as
  - $P=(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where :

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

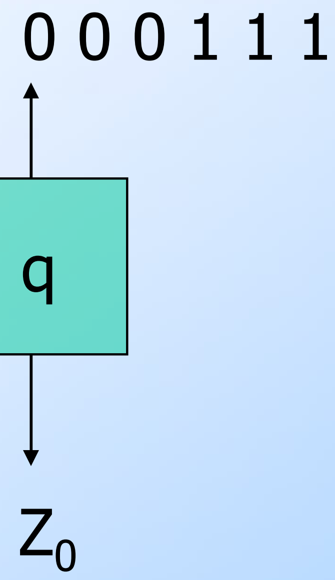
Example: PDA – (contd..)

- 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.

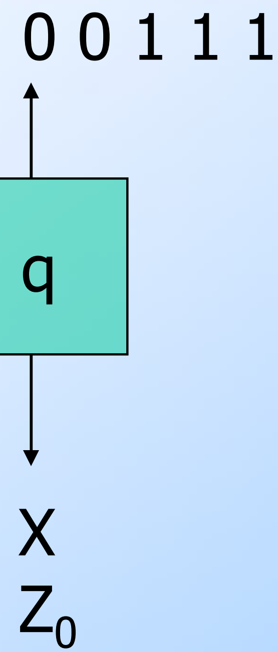
Example: PDA – (3)

- 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.
  - $\delta(q, 1, X) = \{(p, \epsilon)\}$. When we see a 1, go to state p and pop one X.
  - $\delta(p, 1, X) = \{(p, \epsilon)\}$. Pop one X per 1.
  - $\delta(p, \epsilon, Z_0) = \{(f, Z_0)\}$. Accept at bottom.

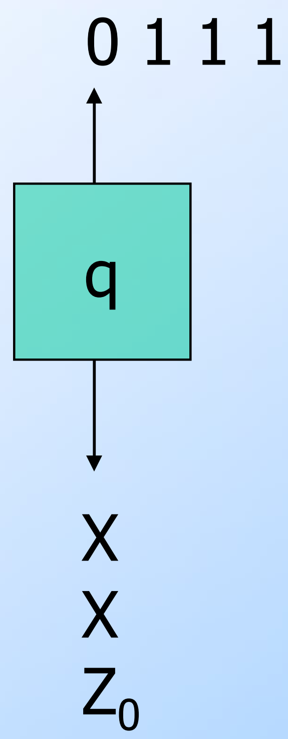
Actions of the Example PDA



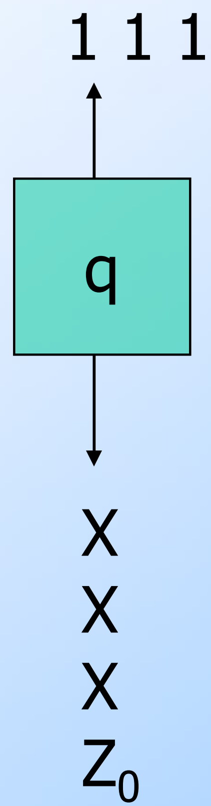
Actions of the Example PDA



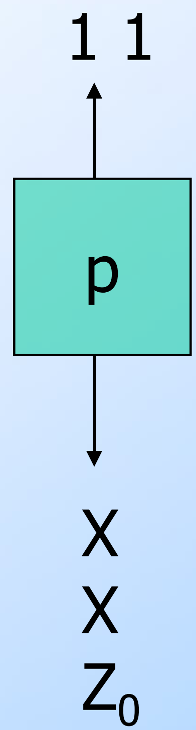
Actions of the Example PDA



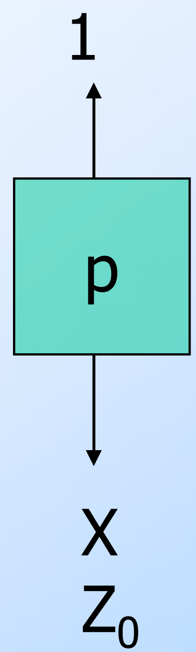
Actions of the Example PDA



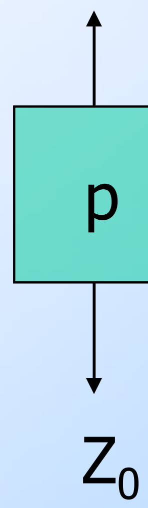
Actions of the Example PDA



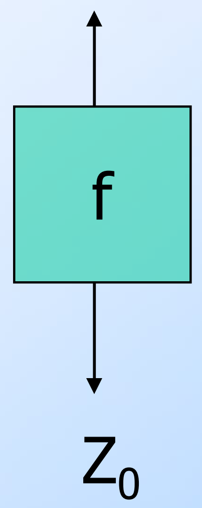
Actions of the Example PDA



Actions of the Example PDA



Actions of the Example PDA



Instantaneous Descriptions

- We can formalize the pictures just seen with an *instantaneous description* (ID).
- The current configuration of a PDA at any instance is described by triplate which is called ID of PDA
- 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:
  $(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash$
  $(q, 0111, XXZ_0) \vdash (q, 111, XXXZ_0) \vdash$
  $(p, 11, XXZ_0) \vdash (p, 1, XZ_0) \vdash (p, \epsilon, Z_0) \vdash$
  (f, ϵ, Z_0)
- Thus, $(q, 000111, Z_0) \vdash^* (f, \epsilon, Z_0)$.
- What would happen on input 0001111?

Answer

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

- $(q, 0001111, Z_0) \vdash (q, 001111, XZ_0) \vdash (q, 01111, XXZ_0) \vdash (q, 1111, XXXZ_0) \vdash (p, 111, XXZ_0) \vdash (p, 11, XZ_0) \vdash (p, 1, Z_0) \vdash (f, 1, Z_0)$
- 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 $(q_0, w, Z_0) \vdash^* (f, \epsilon, \alpha)$ for final state f and any α.

Language of a PDA – (2)

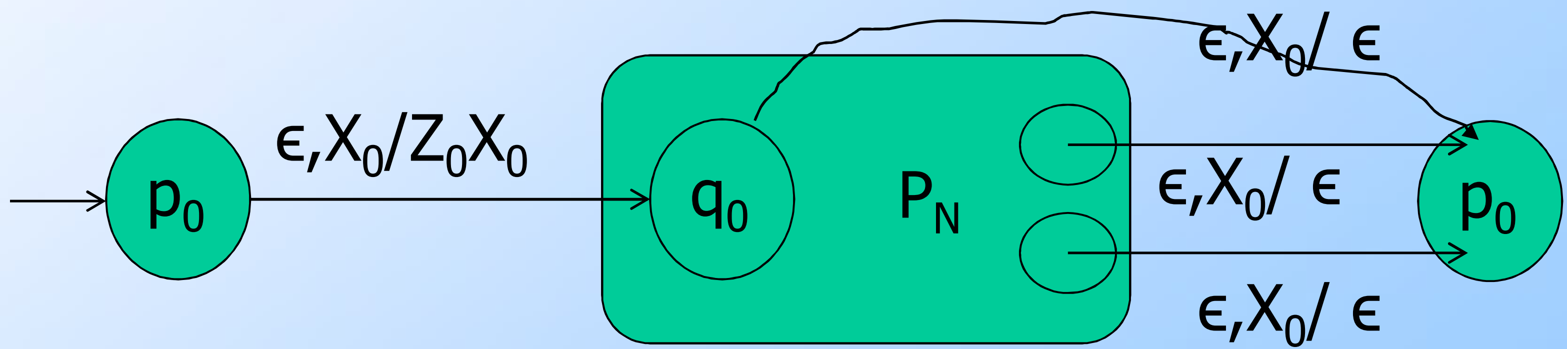
- Another language defined by the same PDA is by *empty stack*.
- If P is a PDA, then N(P) is the set of strings w such that $(q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)$ for any state q.
- In this case , the PDA is defined by 6-tuples only where there is no final states.

PDA from Empty Stack to Final State

- Given a PDA that accepts by empty stack, $P_N=(Q,\Sigma,\Gamma,\delta_N,q_0,Z_0)$.
- Let P_F is PDA accepting by Final State.
- Let X_0 not in Γ , is the start symbol of P_F and use bottom marker of P_N after emptying its stack.
- Define a start state p_0 of $P_{F,}$ and from p_0 it pushes Z_0 on the top of stack and enters into q_0, start state of P_N.
- From q_0, other moves are same for P_F like P_N
- When the stack of P_N becomes empty entering any state p, P_F sees the X_0 on the top of stack.
- Add another state $p_{f,}$ so that from p, P_F moves to p_f which is final state.

Continue……

- The complete specification of P_F is :
  - $P_F=(Q \cup \{p_0,p_f\}, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$ and δ_F is defined by,
    1. **$\delta_F(p_0, \epsilon, X_0)=\{(q_0, Z_0X_0)\}$**
    2. For all state q in Q, a in Σ or a = ϵ, stack symbol Y in Γ, **$\delta_F(q, a, Y)=\delta_N(q, a, Y)$.**
    3. **$\delta_F(q, \epsilon, X_0)=(p_f, \epsilon)$.**



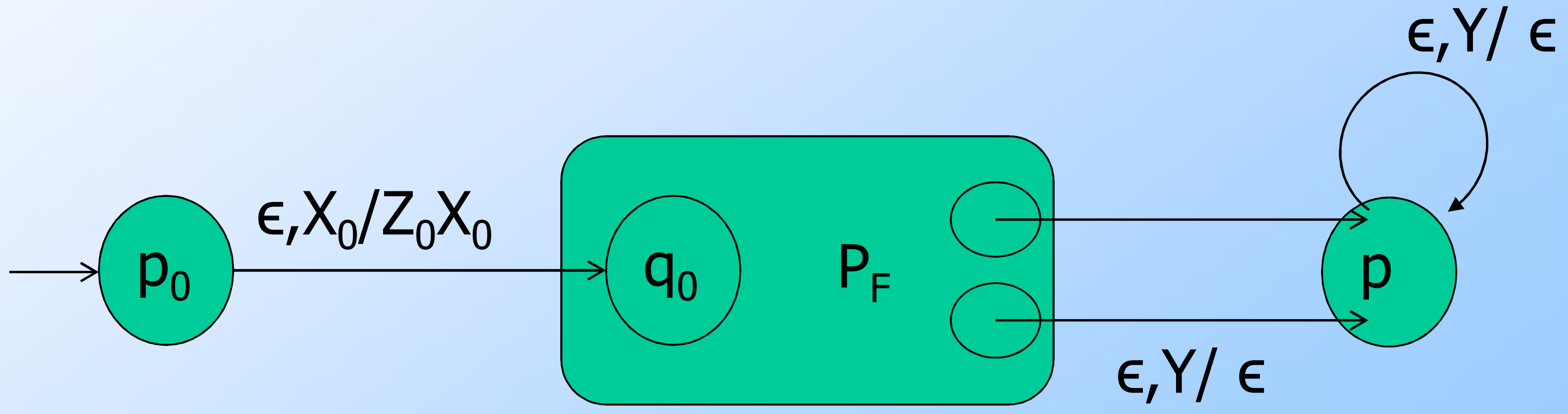
Simulating empty stack by Final State

PDA from Final state to empty Stack

- Given a PDA that accepts by empty stack, $P_F=(Q,\Sigma,\Gamma,\delta_F,q_0,Z_0,F)$. We can construct PDA accepting same language by empty stack as:
  - Let P_N is PDA accepting by empty stack.
  - Let X_0 not in Γ , is the start symbol of P_N and use bottom marker of P_F.
  - Define a start state p_0 of $P_{N,}$ and from p_0 it pushes Z_0 on the top of stack and enters into q_0, start state of P_F on input ϵ.
  - From q_0, other moves are same for P_N like P_F
  - From each final state of P_F , add transition on ϵ to new state p and pop from stack.
  - From state p, pop each stack symbol until it is empty since this is the situation after consuming input string.
  - Thus P_N accepts by empty stack.

Continue……

- The complete specification of P_N is :
  - $P_N=(Q \cup \{p_0,p\}, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$ and δ_N is defined by,
    1. **$\delta_N(p_0, \epsilon, X_0)=\{(q_0, Z_0X_0)\}$**
    2. For all state q in Q, a in Σ or a = ϵ, stack symbol Y in Γ, **$\delta_N(q, a, Y)=\delta_F(q, a, Y)$.**
    3. **For all q in F, Y** in Γ **or Y = X_0 , $\delta_N(q, \epsilon, Y)=(p, \epsilon)$.**
    4. **For all Y** in Γ **$\cup$ $\{X_0\}$, $\delta_N(p, \epsilon, Y)=(p, \epsilon)$.**



Simulating Final state PDA by empty stack PDA

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, $\delta(q, a, X)$ and $\delta(q, \epsilon, X)$ cannot both be nonempty.

Exercise

1. Construct a PDA accepting a language L={w | w is in {a,b}* and w has equal no of a's and b's }
2. Construct a PDA accepting L ={ww^R | w is in {0,1}* }
3. Construct a PDA acceting L ={wcw^R | w is in {1,0}*}