Pushdown Automata

Definition

Moves of the PDA

Languages of the PDA

Deterministic PDA's

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 $\delta(q, a, Z)$ 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.

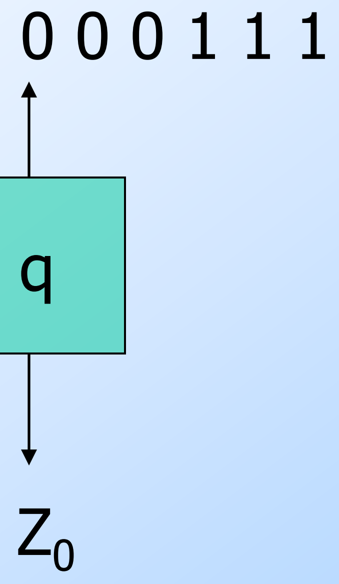
Example: PDA – (2)

- 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.

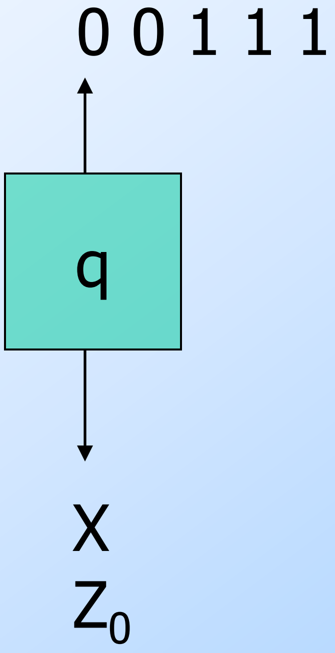
Example: PDA – (3)

- 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.
  - $\delta(q, 1, X) = \{(p, \epsilon)\}$. 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.

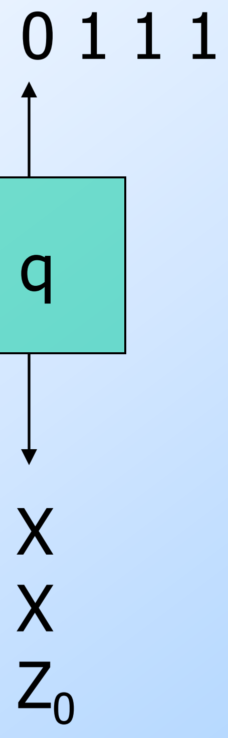
Actions of the Example PDA



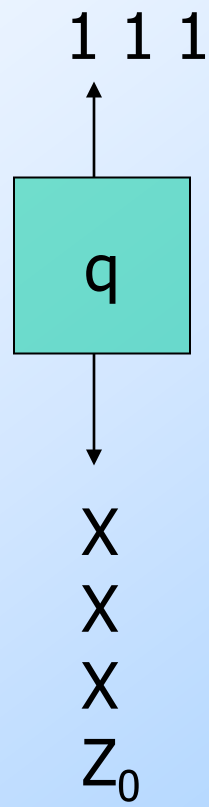
Actions of the Example PDA



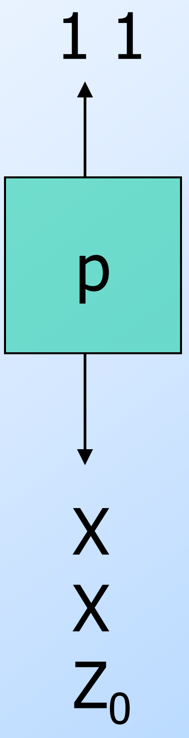
Actions of the Example PDA



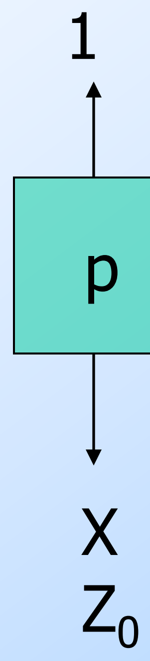
Actions of the Example PDA



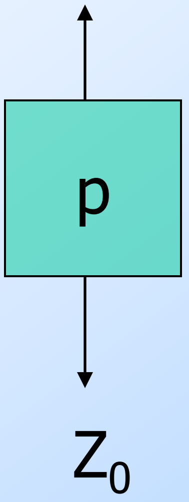
Actions of the Example PDA



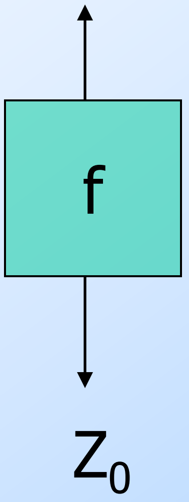
Actions of the Example PDA



Actions of the Example PDA



Actions of the Example PDA



Instantaneous Descriptions

- We can formalize the pictures just seen with an *instantaneous description* (ID).
- 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: $(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash (q, 0111, XXZ_0) \vdash (q, 111, XXXZ_0) \vdash (p, 11, XXZ_0) \vdash (p, 1, XZ_0) \vdash (p, \epsilon, Z_0) \vdash (f, \epsilon, Z_0)$
- 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.

Aside: FA and PDA Notations

- We represented moves of a FA by an extended δ, which did not mention the input yet to be read.
- We could have chosen a similar notation for PDA's, where the FA state is replaced by a state-stack combination, like the pictures just shown.

FA and PDA Notations – (2)

- Similarly, we could have chosen a FA notation with ID's.
  - Just drop the stack component.
- Why the difference? My theory:
- FA tend to model things like protocols, with indefinitely long inputs.
- PDA model parsers, which are given a fixed program to process.

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, ϵ, α) 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.

Equivalence of Language Definitions

1. If L = L(P), then there is another PDA P' such that L = N(P').
2. If L = N(P), then there is another PDA P'' such that L = L(P'').

Proof: L(P) -> N(P') Intuition

- P' will simulate P.
- If P accepts, P' will empty its stack.
- P' has to avoid accidentally emptying its stack, so it uses a special bottom-marker to catch the case where P empties its stack without accepting.

Proof: L(P) -> N(P')

- P' has all the states, symbols, and moves of P, plus:
  1. Stack symbol X_0, used to guard the stack bottom against accidental emptying.
  2. New start state s and "erase" state e.
  3. $\delta(s, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. Get P started.
  4. $\delta(f, \epsilon, X) = \delta(e, \epsilon, X) = \{(e, \epsilon)\}$ for any final state f of P and any stack symbol X.

Proof: N(P) -> L(P'') Intuition

- P'' simulates P.
- P'' has a special bottom-marker to catch the situation where P empties its stack.
- If so, P'' accepts.

Proof: N(P) -> L(P'')

- P'' has all the states, symbols, and moves of P, plus:
  1. Stack symbol X_0, used to guard the stack bottom.
  2. New start state s and final state f.
  3. $\delta(s, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. Get P started.
  4. $\delta(q, \epsilon, X_0) = \{(f, \epsilon)\}$ for any state q of P.

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.