Pushdown Automata

Definition

Moves of the PDA

Languages of the PDA

Deterministic PDA's

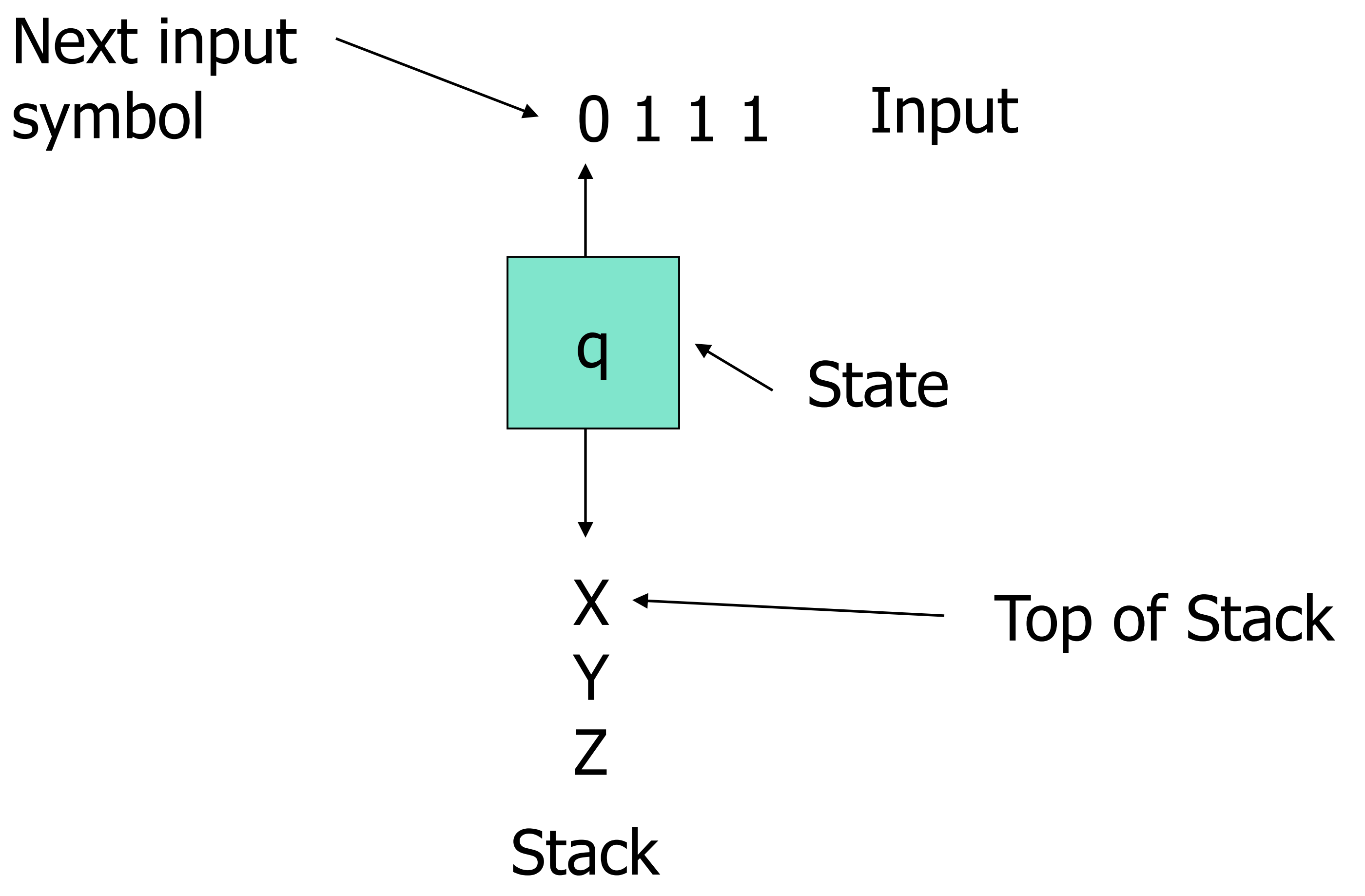
Pushdown Automata

- 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.

Picture of a PDA



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.

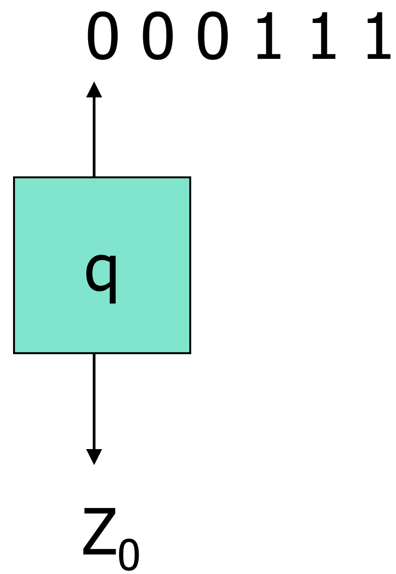
Example: PDA – (2)

- 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.

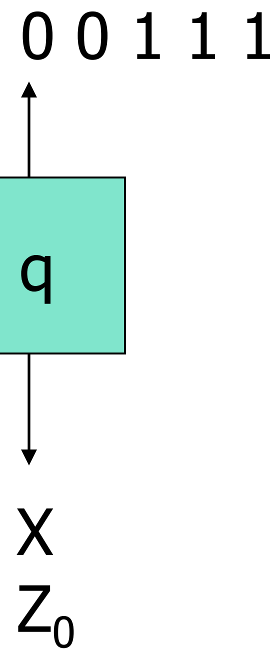
Example: PDA – (3)

- 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.

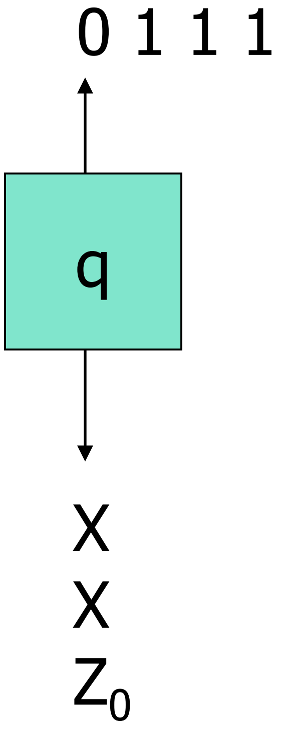
Actions of the Example PDA



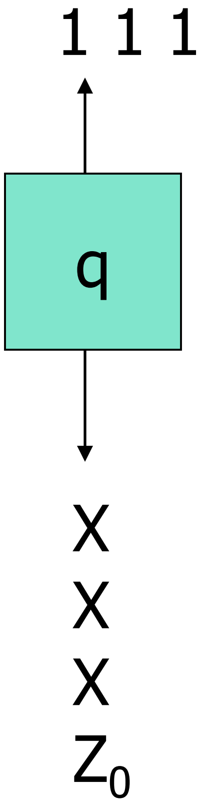
Actions of the Example PDA



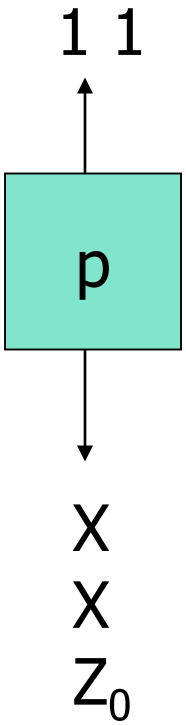
Actions of the Example PDA



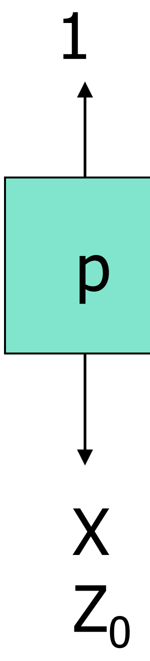
Actions of the Example PDA



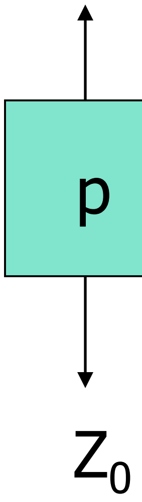
Actions of the Example PDA



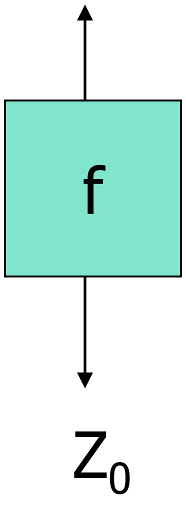
Actions of the Example PDA



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, ϵ, Z_0)
- Thus, $(q, 000111, Z_0) \vdash^* (f, \epsilon, Z_0)$.
- What would happen on input 0001111?

Answer

- $(q, 0001111, Z_0) \vdash (q, 001111, XZ_0) \vdash (q, 01111, XXZ_0) \vdash (q, 1111, XXXZ_0) \vdash (p, 111, XXZ_0) \vdash (p, 11, XZ_0) \vdash (p, 1, Z_0) \vdash (f, 1, Z_0)$
- 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.

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 (the start symbol of P'), used to guard the stack bottom.
  2. New start state s and "erase" state e.
  3. $\delta(s, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. Get P started.
  4. Add $\{(e, \epsilon)\}$ to $\delta(f, \epsilon, X)$ for any final state f of P and any stack symbol X, including X_0.
  5. $\delta(e, \epsilon, X) = \{(e, \epsilon)\}$ for any 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 (the start symbol), 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, δ(q, a, X) and δ(q, ϵ, X) cannot both be nonempty.

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.
- Example: It is easy to see how a PDA can recognize balanced parentheses; not so easy as a grammar.

Converting a CFG to a PDA

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

Intuition About P

- 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, α) for any x if and only if S $\Rightarrow^*_{lm}$ wα.
- Part 1: "only if" is an induction on the number of steps made by P.
- Basis: 0 steps.
  - Then α = S, w = ϵ, and S $\Rightarrow^*_{lm}$ S is surely true.

Induction for Part 1

- Consider n moves of P: $(q, wx, S) \vdash^* (q, x, \alpha)$ and assume the IH for sequences of n-1 moves.
- There are two cases, depending on whether the last move uses a Type 1 or Type 2 rule.

Use of a Type 1 Rule

- The move sequence must be of the form $(q, yax, S) \vdash^* (q, ax, a\alpha) \vdash (q, x, \alpha)$, where ya = w.
- By the IH applied to the first n-1 steps, $S \Rightarrow^*_{lm} ya\alpha$.
- But ya = w, so $S \Rightarrow^*_{lm} w\alpha$.

Use of a Type 2 Rule

- The move sequence must be of the form $(q, wx, S) \vdash^* (q, x, A\beta) \vdash (q, x, \gamma\beta)$, where $A \to \gamma$ is a production and $\alpha = \gamma\beta$.
- By the IH applied to the first n-1 steps, $S \Rightarrow^*_{lm} wA\beta$.
- Thus, $S \Rightarrow^*_{lm} w\gamma\beta = w\alpha$.

Proof of Part 2 ("if")

- We also must prove that if $S =>*_{lm} w\alpha$, then $(q, wx, S) \vdash* (q, x, \alpha)$ for any x.
- Induction on number of steps in the leftmost derivation.
- Ideas are similar; omitted.

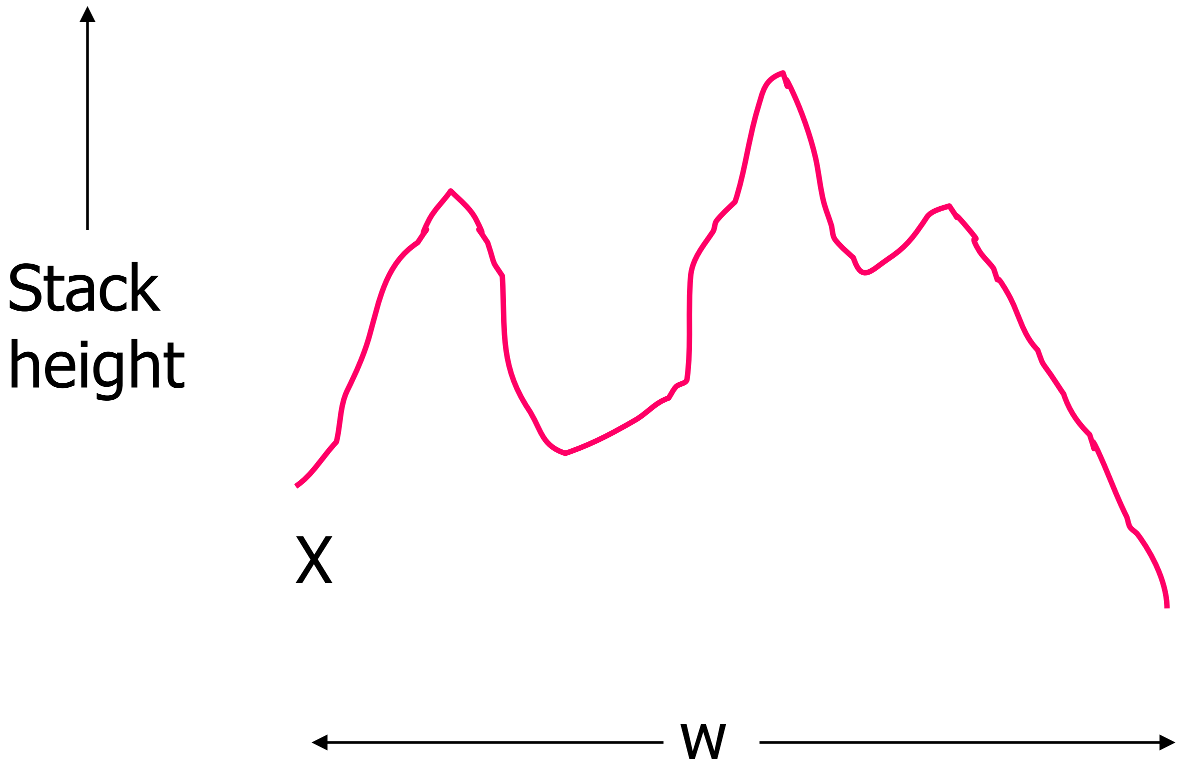
Proof – Completion

- We now have $(q, wx, S) \vdash^* (q, x, \alpha)$ for any x if and only if $S \Rightarrow^*_{lm} w\alpha$.
- In particular, let $x = \alpha = \epsilon$.
- Then $(q, w, S) \vdash^* (q, \epsilon, \epsilon)$ if and only if $S \Rightarrow^*_{lm} w$.
- That is, w is in N(P) if and only if w is in L(G).

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 [pXq] generating exactly the inputs that cause P to have the net effect of popping stack symbol X while going from state p to state q.
  - P never gets below this X while doing so.

Picture: Popping X



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.

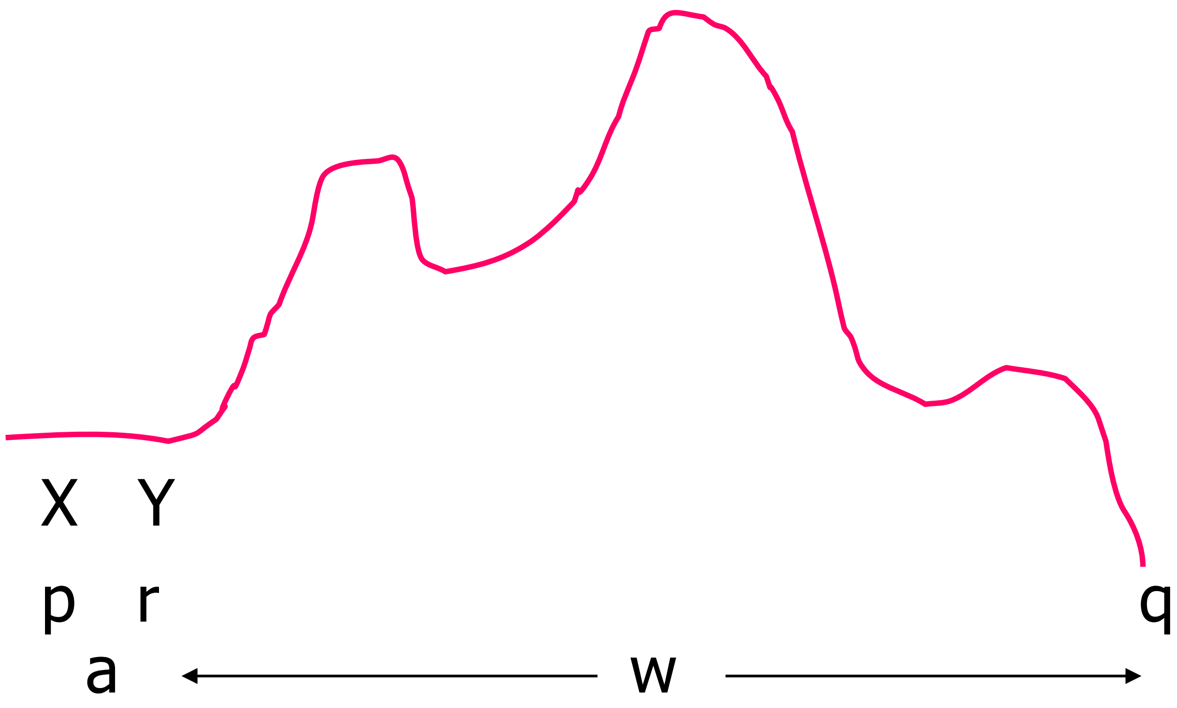
Productions of G

- 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).
  - Note *a* can be an input symbol or ϵ.
- Then the production is [pXq] -> a.
- 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: δ(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.

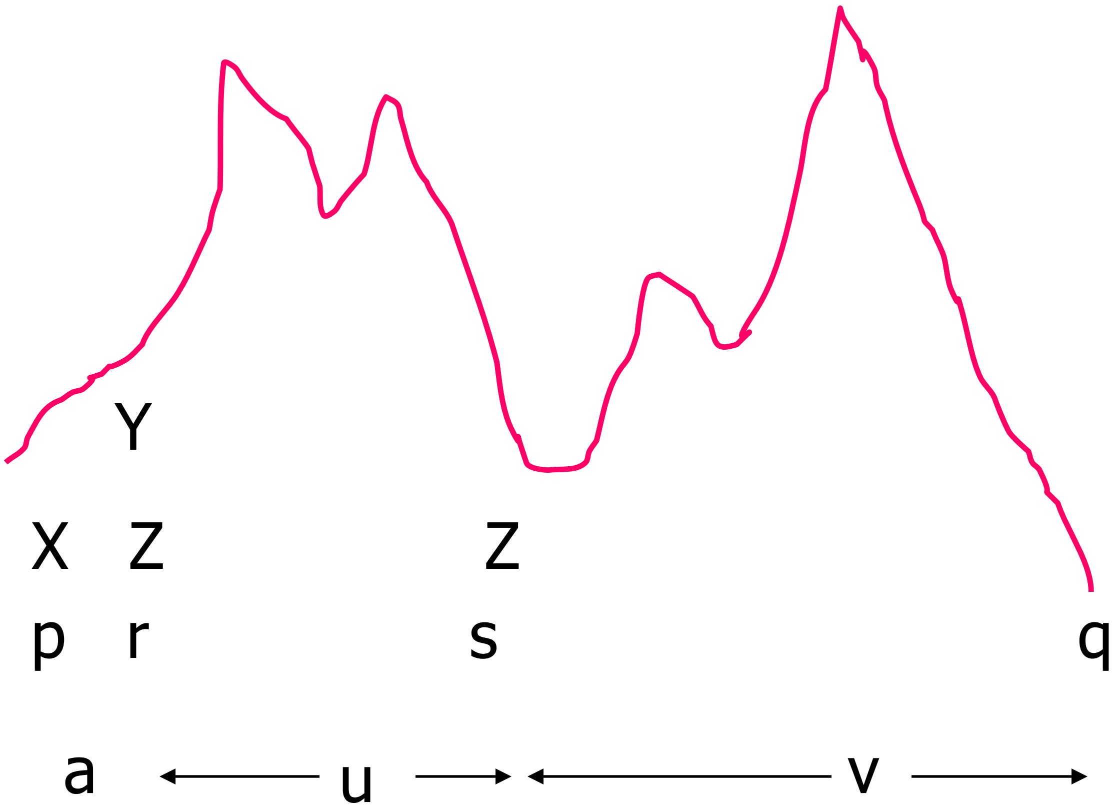
Picture of the Action



Productions of G – (3)

- Third simplest case: δ(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 the Action



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] =>* auv whenever [rYs] =>* u and [sZq] =>* v.

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

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

Completion of the Construction

- We can prove that $(q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)$ if and only if $[q_0Z_0p] \Rightarrow^* w$.
  - Proof is two easy inductions.
- But state p can be anything.
- Thus, add to G another variable S, the start symbol, and add productions $S \rightarrow [q_0Z_0p]$ for each state p.