Grammar & Pushdown automata

- Grammars
- Pushdown automata (pda): nfa + stack
- one-state pda
- context free grammar has the power as pda.

Grammars

$V = \{ S \}, \ \Sigma = \{ 0,1\}$

$S \rightarrow 0S1$

$S \rightarrow 01$

$\Rightarrow$

Language:

$\{ 01, 0011, 000111, \ \ldots\}$

A grammar G is a 4-tuple (V, Σ, R, S), where

- V is a finite set called the variables (or non-terminals).
- Σ is the alphabet, its elements are also called terminals. Note that Σ and V are disjoint.
- S, which is an element in V, is the start variable.
- R is finite of rules (productions).

What is a rule (production)? A rule takes the form

$z_1 \rightarrow z_2$, where z_1 and z_2 are strings (words) over $V \cup \Sigma$ and z_1 must contain at least one variable.

Example

$V = \{ S \}$, $\Sigma = \{ 0,1\}$, R contains two rules:

$S \rightarrow 0S1$

$S \rightarrow 01$

Given a grammar, we are interested in the words over Σ that are derived from the starting symbol (i.e. S). Roughly speaking, S derives a word w if by applying the rules repeatedly, we can eventually obtain w.

Example

$V = \{ S \}$, $\Sigma = \{ 0,1\}$, R contains two rules:

$S \rightarrow 0S1$

$S \rightarrow 01$

Given a grammar, we are interested in the words over Σ that are derived from the starting symbol (i.e. S). Roughly speaking, S derives a word w if by applying the rules repeatedly, we can eventually obtain w.

Example: S derives 000111.

Derivation process: $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000111$.

00011 cannot be derived from S.

In general, S derives 0^i1^i for any integer $i \geq 1$.

Derivation

Let $G = (V, \Sigma, R, S)$ be a grammar. Consider a rule $z_1 \rightarrow z_2$.

Let W_1 be a word over $(V \cup \Sigma)^*$ containing z_1 as a substring.
E.g, $W_1 = ab\, z_1\, cd$

Using the rule $z_1 \rightarrow z_2$, we transform W_1 to another word W_2 by replacing z_1 with z_2. E.g., $W_2 = ab\, z_2\, cd$.

In this case, we say that W_1 yields (or directly derives) W_2.
Notation: $W_1 \Rightarrow W_2$

Derivation

Let $G = (V, \Sigma, R, S)$ be a grammar. Consider a rule $z_1 \rightarrow z_2$.

Let W_1 be a word over $(V \cup \Sigma)^*$ containing z_1 as a substring.
E.g, $W_1 = ab\, z_1\, cd$

Using the rule $z_1 \rightarrow z_2$, we transform W_1 to another word W_2 by replacing z_1 with z_2. E.g., $W_2 = ab\, z_2\, cd$.

In this case, we say that W_1 yields (or directly derives) W_2.
Notation: $W_1 \Rightarrow W_2$

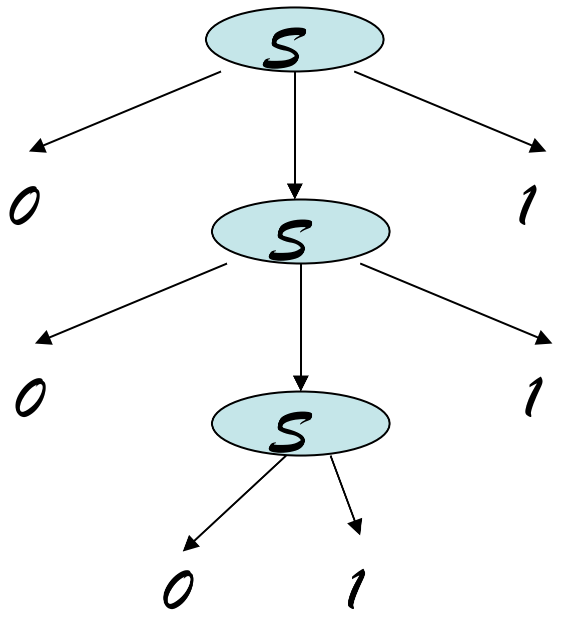
A word W derives another word W' (notation: $W \overset{*}{\Rightarrow} W'$) if

- $W=W'$ or
- there exists $k \geq 0$ words $W_0, W_1, \ldots, W_{k-1}$ such that $W \Rightarrow W_0 \Rightarrow W_1 \Rightarrow W_2 \Rightarrow \ldots \Rightarrow W_{k-1} \Rightarrow W'$

The language of G, denoted by L(G), is $\{ W \in \Sigma^* \mid S \overset{*}{\Rightarrow} W \}$.

Parse Tree

- Given a string w in L(G). The way w is derived from the start symbol can be represented by a tree.
- E.g., w = 000111



Example

V = {S}, $\sum$ = { 0,1}, R contains two rules:

S $\rightarrow$ 0S1

S $\rightarrow$ 01

The language generated by the above grammar is

{ 0^i1^i | $i \geq 1$ }.

Give a grammar to generate { $0^n1^n2^n$ | $n \geq 1$ }.

Example

Give a grammar to generate $\{ 0^n 1^n 2^n \mid n \geq 1 \}$.

$S \to 0SAB$

$S \to 0AB$

$BA \to AB$

$0A \to 01 \qquad 1A \to 11$

$1B \to 12 \qquad 2B \to 22$

Linear and Context-Free Grammars

- A context-free grammar can have rules only of the form $A \rightarrow Z$ where A is a variable and Z is a word over $V \cup \Sigma$. E.g., $A \rightarrow 0A1B$
- A right linear grammar can have rules only of the form $A \rightarrow Z$ where A is a variable and Z = aB or a, where a is a terminal and B is variable.

  [NB. left linear grammar: Z = Ba or a.]
- Right linear grammar ≡ finite automata
- Context free grammar ≡ pushdown automata (finite automata with a stack)

Pushdown Automata (pda)

Roughly speaking, pda = nfa + stack.

Push & Pop;
Last in first out

How does a pda operate?

In each step, a pda reads a symbol of input & *pops the top symbol from stack*;

Depending on the input symbol, stack symbol & current state, the pda changes its state & *pushes a string onto the stack*.

ε-move is allowed (i.e., make a move without reading a symbol of input).

What happens when stack is empty? It halts.

Formal definition

A pushdown automaton is a 7-tuple $(Q, \Sigma, \Gamma, f, q_0, s_0, F)$.

- Q is a finite set of states.
- Σ is the input alphabet.
- Γ is the stack alphabet.
- f is the transition function. For any $q \in Q$, $a \in \Sigma$, $s \in \Gamma$, $f(q, a, s) = \{ (q_1, z_1), (q_2, z_2), \ldots \}$, where each $z_i \in \Gamma^*$.

  Formally, $f : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \rightarrow P(Q \times \Gamma^*)$
- q_0 is the start state; s_0 is the initial stack symbol.
- $F \subseteq Q$ is the set of accept states.

NB. Sipser's book: $s_0 = \varepsilon$; f: $Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \cup \{\varepsilon\} \rightarrow \ldots$.

Configurations

A pda starts in state q_0 and with s_0 in the stack.

After operating for a while, the configuration of a pda can be characterized by 3 components:

- current state, $q \in Q$
- remaining input, $w_i\ w_{i+1} \ldots w_n \in \Sigma^*$ (w_i is the next input symbol)
- entire stack content, $s_1 s_2 \ldots s_m \in \Gamma^*$ (s_m is the top of stack)

Next input symbol

Top of stack

$(q, w_i\ w_{i+1} \ldots w_n, s_1 s_2 \ldots s_m)$ is called a configuration of a pda.

Initial configuration of a pda with input w: (q_0, w, s_0)

Acceptance of pda

Suppose that $f(q, a, s) = \{ \ldots., (q',z), \ldots\}$.

Then for any w in Σ and z′ in Γ^*,

the configuration $(q, aw, z's)$ can move to

the configuration $(q', w, z'z)$ in one step.

Notation: $(q, aw, z's) \Rightarrow (q', w, z'z)$

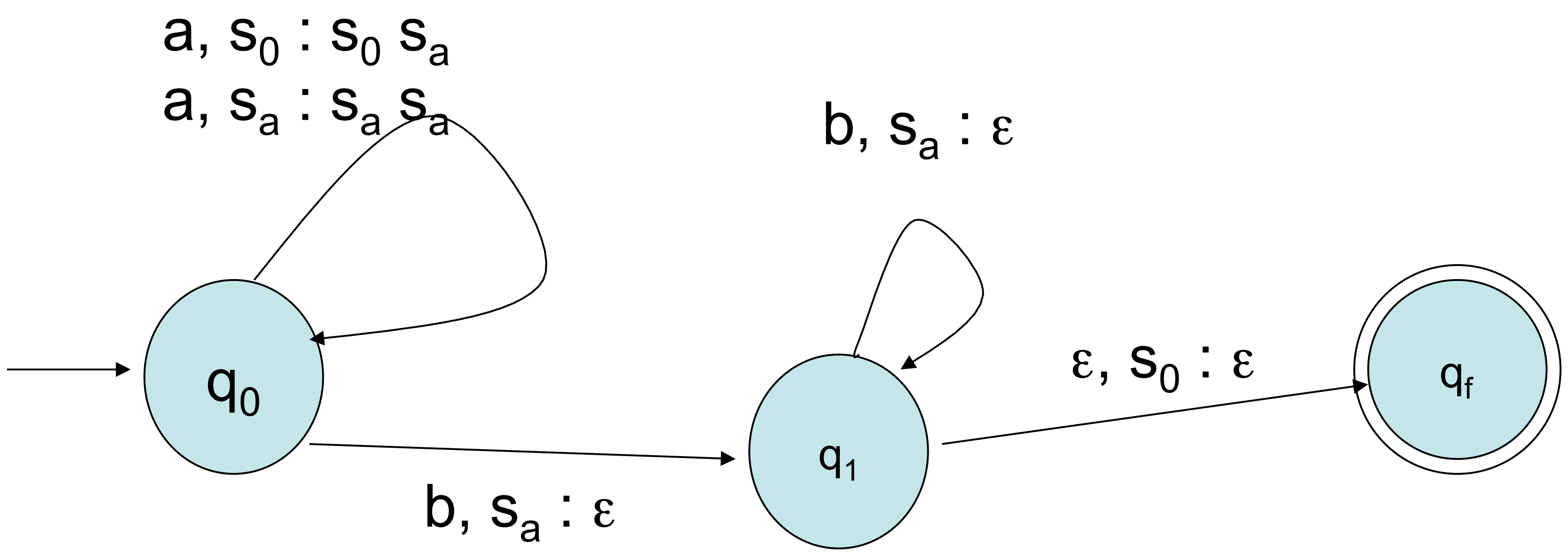
Accepting configuration: $(q_f, \varepsilon, \varepsilon)$, where $q_f \in F$.

A pda M accepts an input w if $(q_0, w, s_0) \Rightarrow$ an accepting configuration.

$L(M) = \{ w \mid M \text{ accepts } w \}$.

NB. Sipser: accepting configuration can have a non-empty stack.

Example: A pda for $\{a^n b^n \mid n > 0\}$



Stack symbols: s_0, s_a

$f(q_0, a, s_0) = \{ (q_0, s_0 s_a)\}$, $f(q_0, a, s_a) = \{ (q_0, s_a s_a)\}$
$f(q_0, b, s_a) = \{(q0, \varepsilon)\}$
$f(q_1, b, s_a) = \{ (q_1, \varepsilon) \}$
$f(q_1, \varepsilon, s_0) = \{ (q_f, \varepsilon) \}$

Example

On input aabb:

$(q_0, aabb, s_0) \Rightarrow (q_0, abb, s_0 s_a) \Rightarrow (q_0, bb, s_0 s_a s_a) \Rightarrow (q_1, b, s_0 s_a) \Rightarrow$
$(q_1, \varepsilon, s_0) \Rightarrow (q_f, \varepsilon, \varepsilon)$ accepting configuration

On input aaabb:

$(q_0, aaabb, s_0) \Rightarrow (q_0, aabb, s_0 s_a) \Rightarrow (q_0, abb, s_0 s_a s_a) \Rightarrow (q_0, bb,$
$s_0 s_a s_a s_a) \Rightarrow (q_1, b, s_0 s_a s_a) \Rightarrow (q_1, \varepsilon, s_0 s_a)$ not an accepting configuration.

On input aabbb:

$(q_0, aabbb, s_0) \Rightarrow (q_0, abbb, s_0 s_a) \Rightarrow (q_0, bbb, s_0 s_a s_a) \Rightarrow (q_1, bb,$
$s_0 s_a) \Rightarrow (q_1, b, s_0) \Rightarrow (q_f, b, \varepsilon)$ not an accepting configuration.

How powerful is the stack?

Theorem For any pda *A*, there exists a pda B with one state such that L(B) = L(A).

Idea: store the state information in the stack.

Let *A* = $(Q, \Sigma, \Gamma, f, q_0, s_0, F)$.

Construct a pda B as follows:

- B has only one state p.
- Push *A*'s current state into B's stack. I.e., B's stack alphabet $\Gamma_B = Q \times \Gamma$.

How powerful is the stack?

Theorem For any pda A, there exists a pda B with one state such that L(B) = L(A).

Idea: store the state information in the stack.

Let $A = (Q, \Sigma, \Gamma, f, q_0, s_0, F)$.

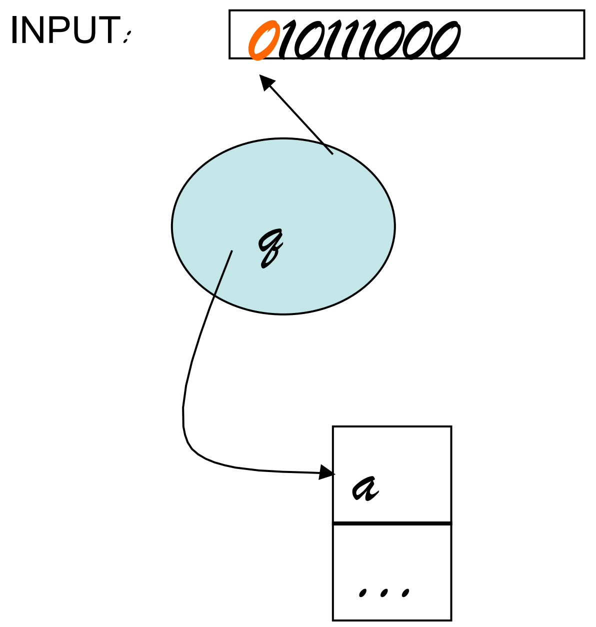
Construct a pda B as follows:

- B has only one state p.
- Push A's current state into B's stack. I.e., B's stack alphabet $\Gamma_B = Q \times \Gamma$.

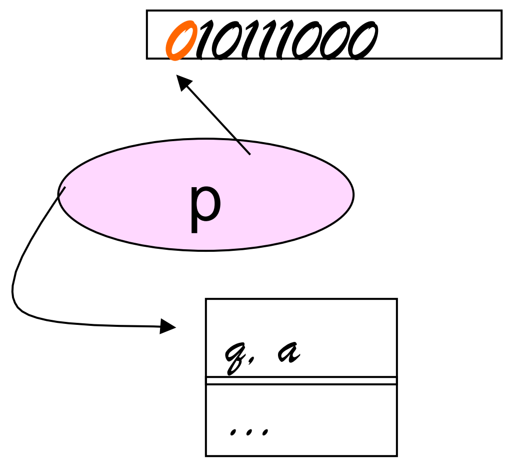
When $[q,s] \in \Gamma_B$ is on top of B's stack, it means that A's current state is q and its top stack symbol is s.

Initial stack symbol of B = $[q_0 , s_0]$.

pda A



One-state
pda B



First attempt

Theorem: For any pda ***A***, there exists a pda **B** with one state such that L(B) = L(A).

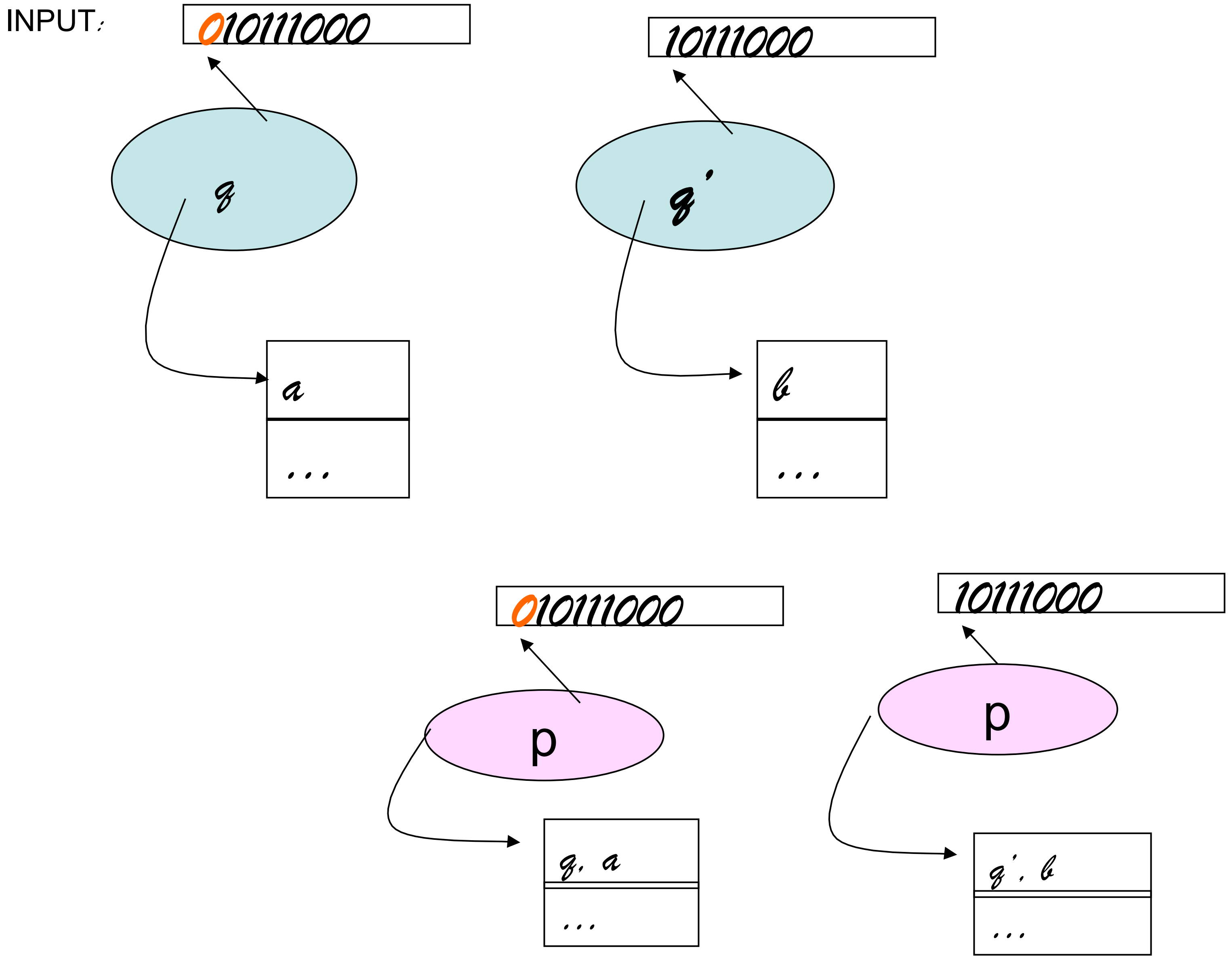
$A = (Q, \Sigma, \Gamma, f, q_0, s_0, F)$.
Suppose $(q', z) \in f(q, a, s)$, where $z = s_1 s_2 \ldots s_m$.
I.e., A will go to state q' and push $s_1 s_2 \ldots s_m$.

Then what should B do? Assume **p** is the only state of B.
Top of stack = [q,s].

$(p, [q', s_1][q', s_2] \ldots [q', s_m]) \in f_B(p, a, [q,s])$

Note that the top symbol on B's stack is $[q', s_m]$.

Example: (q', b) ∈ f (q, 0, a)



Boundary case

Suppose $(q', \varepsilon) \in f(q, a, s)$. I.e., no follow-up push.

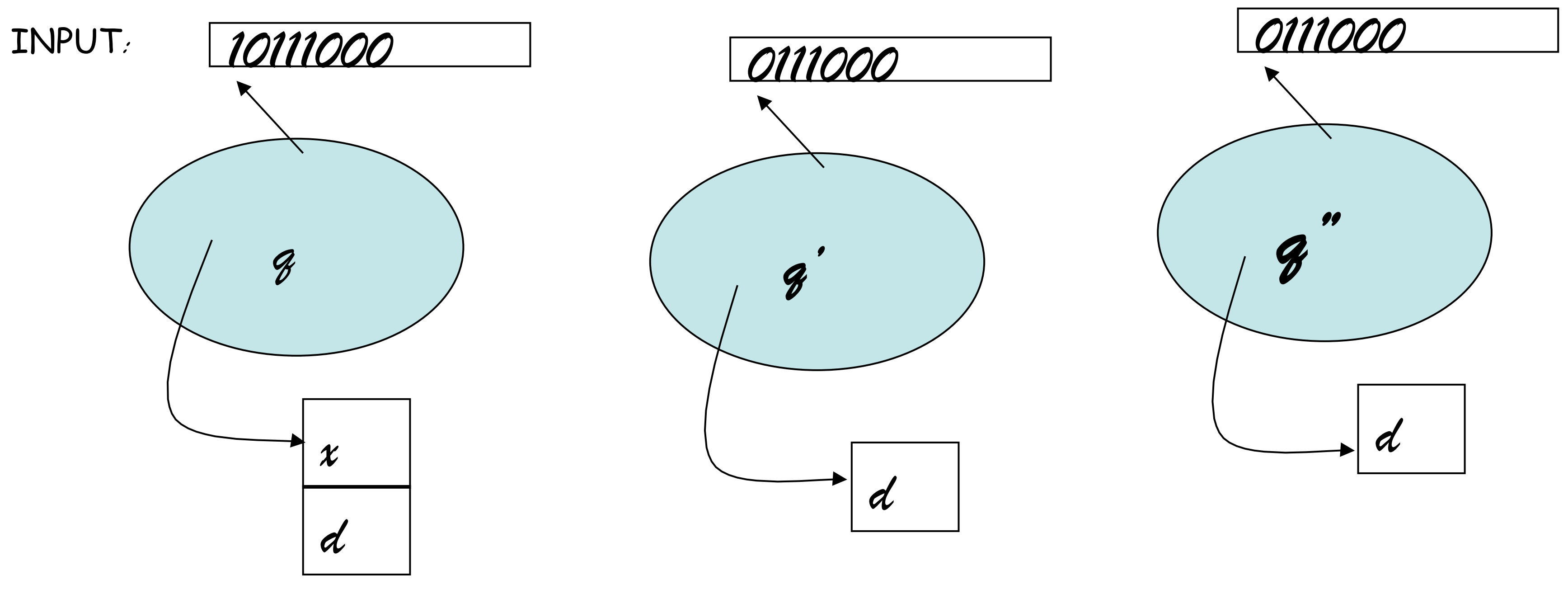
Then what should B do? Recall that B has only one state p.

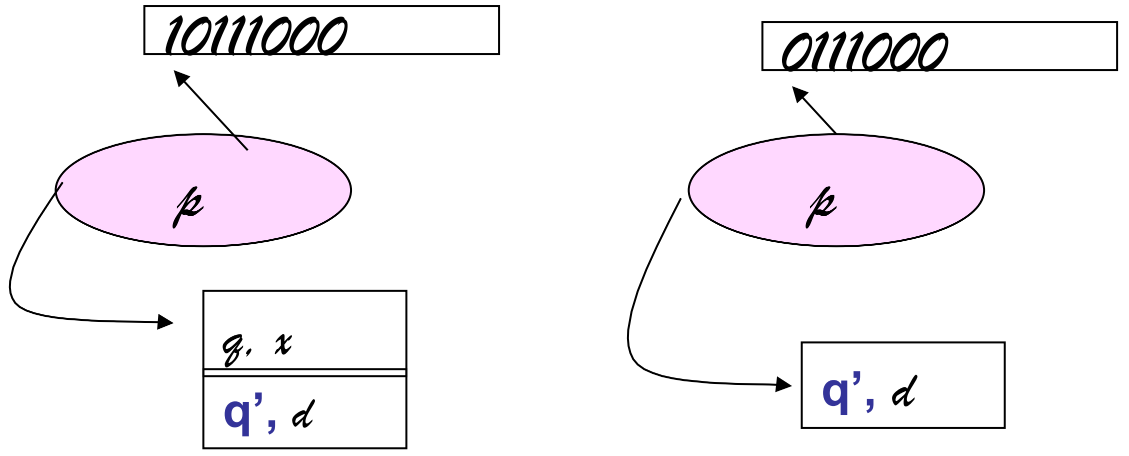
$(p, \varepsilon) \in f_B(p, a, [q,s])$

In this case, both A and B have nothing to push to the stack. I.e., B can't push q' into the stack!

Suppose $(q', \varepsilon), (q'', \varepsilon) \in f(q, a, s)$. How can we obtain two possible configurations with $[q', ?]$ and $[q'', ?]$ at the top of the stack, respectively?

Example: (q', ε), (q'', ε) ∈ f (q, 1, x)



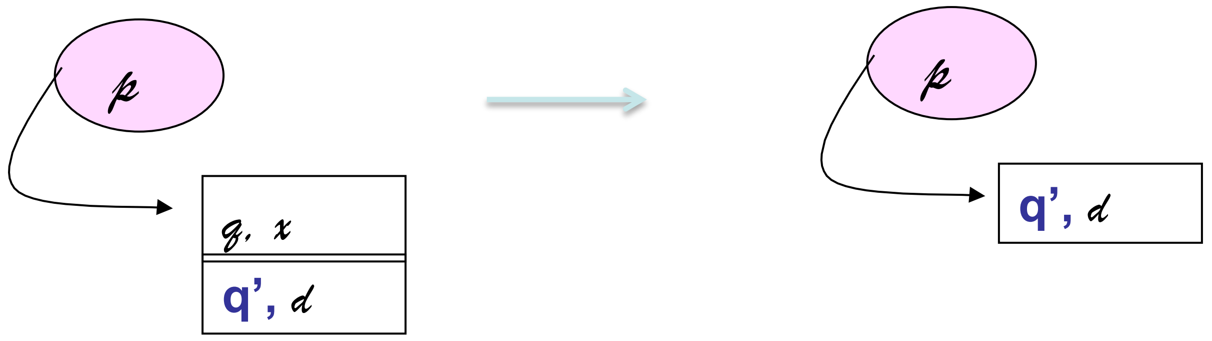


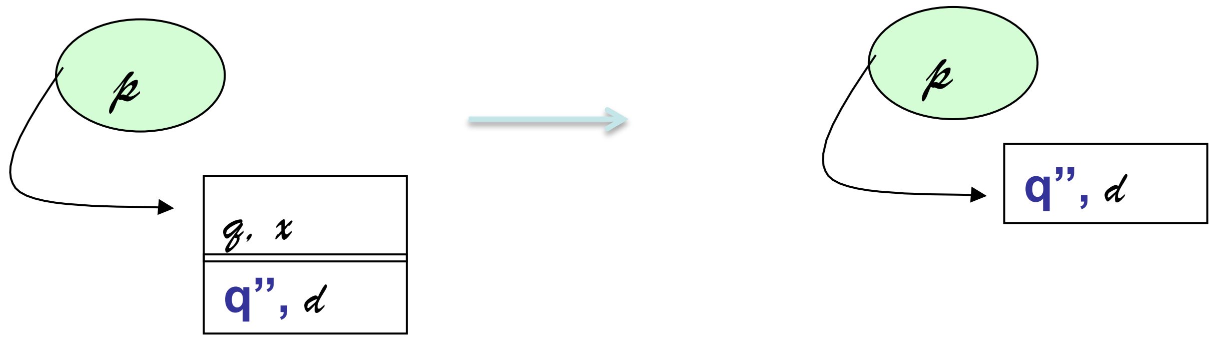
We can't get two different Configurations.

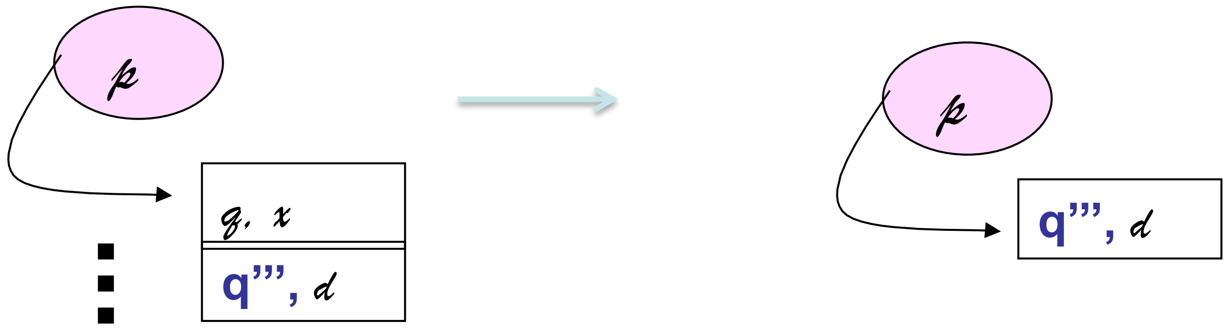
How to fix the problem

Use the nondeterministic power of pda to guess “in advance” what would be the new state after an ε-push.

Example: (q', ε), (q'', ε) ∈ f (q, 1, x)

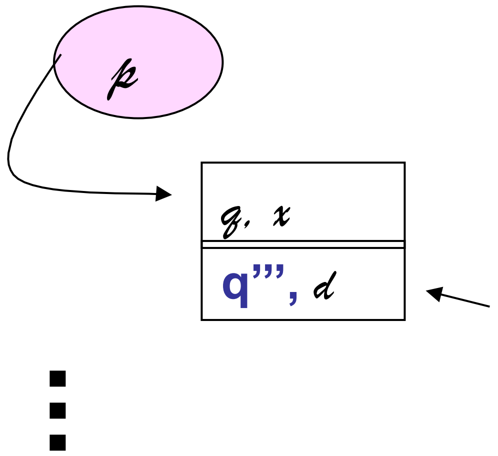


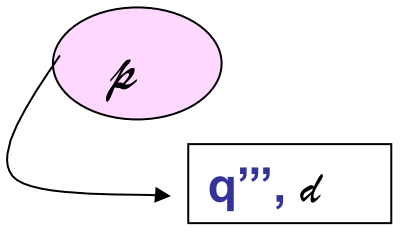




No, we don't want this move!

Example: (q', ε), (q'', ε) ∈ f (q, 1, x)





No, we don't want this move!

How to stop the pda here?

Check if the state of the 2nd-to-top stack entry is q' or q''. Stop if it isn't

Problem: we can only look at the top entry of the stack.

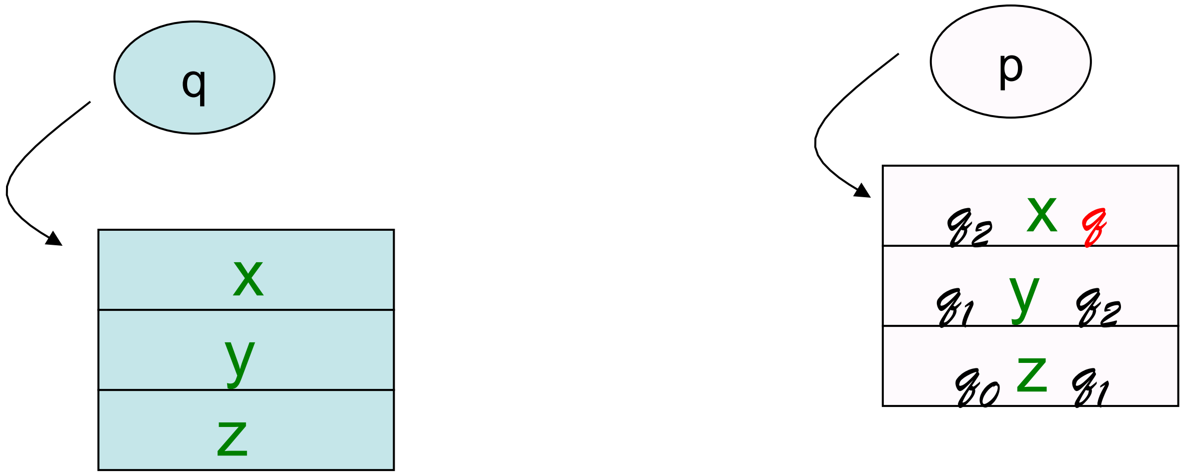
Solution: Copy the state information to the top element of stack.

Guess & Check

Let $\Gamma_B = Q \times \Gamma \times Q$.

When [q', a, q] is at the top of stack, it means that

- a is at the top of A's stack,
- q is the <u>current state</u> of A, and
- q' is the new state stored in the next lower stack entry.



If $(q_2, \varepsilon) \in f(q, 0, a)$, then

$f_B(p, 0, [q_2, a, q])$ contains (p, ε).

Definition

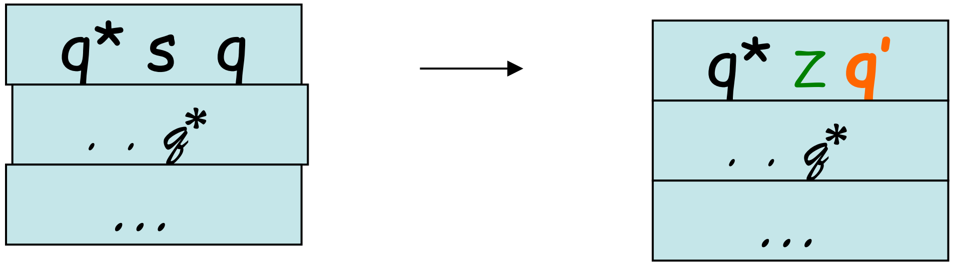
Suppose $(q', z) \in f(q, a, s)$, where z is a single symbol in Γ.
Then what should B do?

For all $q^* \in Q$,

$f_B(p, a, [q^*, s, q])$ contains the move $(p, [q^*, z, q'])$.

NB.

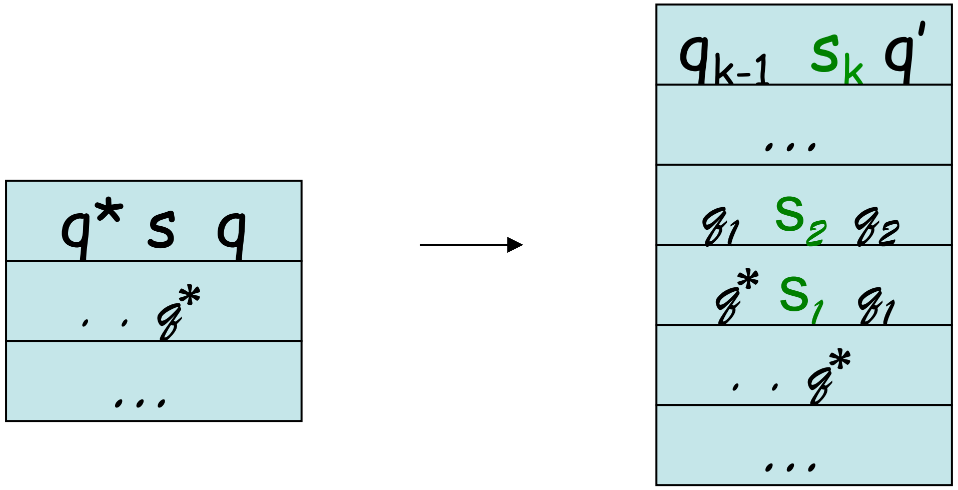
For all … Recall that a pda is nondeterministic in nature.



Definition

Suppose $(q', z) \in f(q, a, s)$, where $z = s_1 s_2 \ldots s_{k-1} s_k$. Then what should B do?

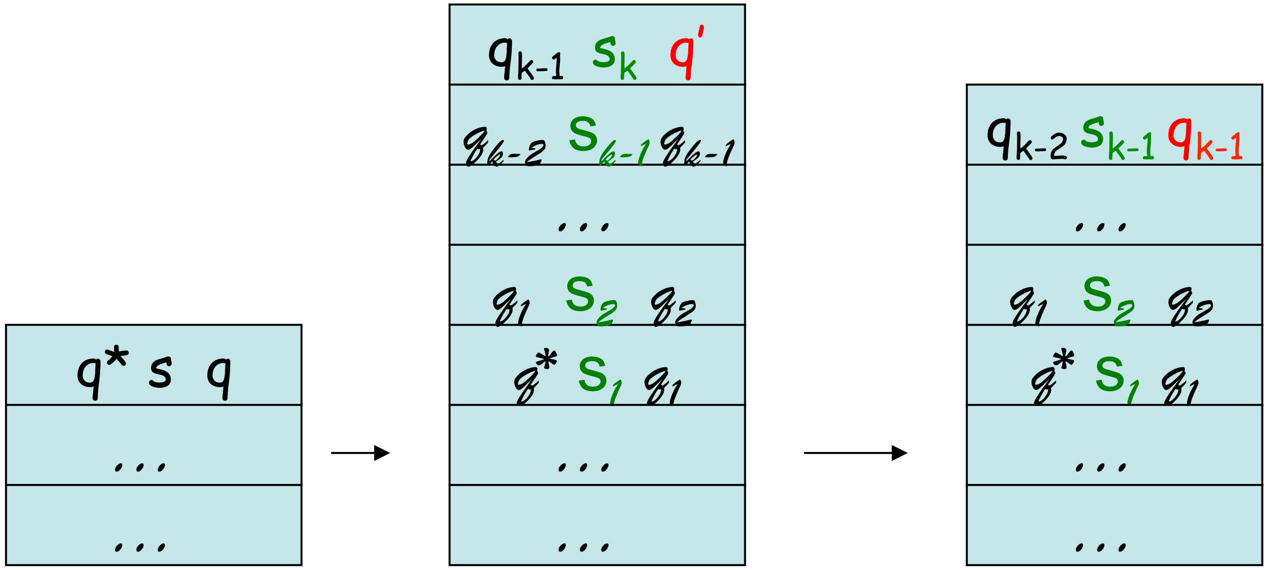
For **all** $q^*, q_1, q_2, \ldots, q_{k-1} \in Q$, $f_B(p, a, [q^*, s, q])$ contains the move $(p, [q^*, s_1, q_1][q_1, s_2, q_2]\ldots[q_{k-2}, s_{k-1}, q_{k-1}][q_k, s_k, q'])$.



ε-push forces B to check its guess

If $(q', \varepsilon) \in f(q, a, s)$, then
  $f_B(p, a, [q', s, q])$ contains (p, ε).

- Example: $(q_{k-1}, \varepsilon) \in f(q', a, s_k)$.



Initial stack symbol

If $F = \{ q_f \}$. Then B's initial stack symbol = $[q_f , s_0, q_0]$.

If F contains more than one state,
Let q# be a new state (i.e., not in Q).
B's initial stack symbol = $[q\# , s_0, q_0]$.

First move of B (nondeterministic):
$f(p, \varepsilon, [q\#, s_0, q_0]) = \{ (p, [q_f, s_0, q_0]) \mid q_f \text{ is in } F \}$

By induction on length of w

Claim: For any input $w \in \Sigma^*$,

$(q_0, w, s_0) \underset{A}{\overset{*}{\Rightarrow}} (q, a, s) \underset{A}{\Rightarrow} (q_f, \varepsilon, \varepsilon)$, where $a \in \Sigma \cup \{\varepsilon\}$ and $s \in \Gamma$

if and only if

$(p, w, [q_f, s_0, q_0]) \underset{B}{\overset{*}{\Rightarrow}} (p, a, [q_f, s, q]) \underset{B}{\Rightarrow} (p, \varepsilon, \varepsilon)$

Context free grammar

Theorem. Let L be a context free language, then there is a pda A such that L(A) = L.

Suppose L = L(G) for some cfg G = (V, Σ, S, R).

We want a pda A such that for any input w $\in$ L, the way A accepts w simulates the way S derives w (specifically, the leftmost derivation of w).

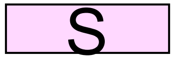
Idea: Use the stack to store the variables to be expanded and the terminals to be matched.

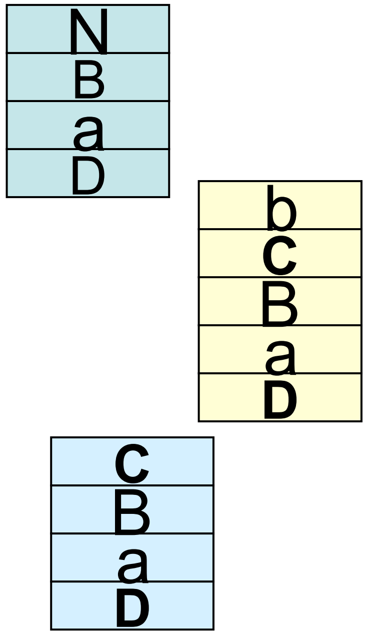
Example

Assume $w = w_1w_2\ldots w_n$.

Suppose $S \to N\ B\ a\ D$, $N \to b\ C$, $C \to d$ are rules in R.

The pda A starts with S in its stack.





Step 1: pop S and push DaBN (to simulate the rule $S \to N\ B\ a\ D$).

Step 2: pop N and push C b (to simulate the rule $N \to b\ C$).

Step 3: pop b from the stack if $b = w_1$.

Step 4: pop C and push d (to simulate the rule $C \to d$).

Definition

Given $G = (V, \Sigma, S, R)$, define

pda $A = (\{q\}, \Sigma, V \cup \Sigma , f, q, S, \{q\})$.

Two types of transition:

- Simulating a rule:

  If R contains a rule $N \rightarrow z$, where $z \in (V \cup \Sigma)^*$.

  then $f(q, \varepsilon, N)$ contains (q, z^T).

- Reading an input symbol:

  $f(q, a, a) = (q, \varepsilon)$

NB. z^T dentoes the transpose of z. E.g., $(abc)^T = cba$

The Invariant

Claim: For any u,v $\in \Sigma^*$ and z $\in (V \cup \Sigma)^*$.

$S \overset{*}{\Rightarrow} u\ z$ if and only if $(q, uv, S) \overset{*}{\Rightarrow} (q, v, z^T)$.

(Proof: by induction on the length of derivation.)

In particular, for any w $\in \Sigma^*$, $S \overset{*}{\Rightarrow} w$ if and only if $(q, w, S) \overset{*}{\Rightarrow} (q, \varepsilon, \varepsilon)$.

Pushdown Automata & CFG

Theorem Let A be a pda such that $|Q| = 1$. Then $L(A)$ is a context free language (i.e., $L(A) = L(G)$ for some context free grammar).

Let $A = (\{q\}, \Sigma, \Gamma, f, q, s_0, \{q\})$.

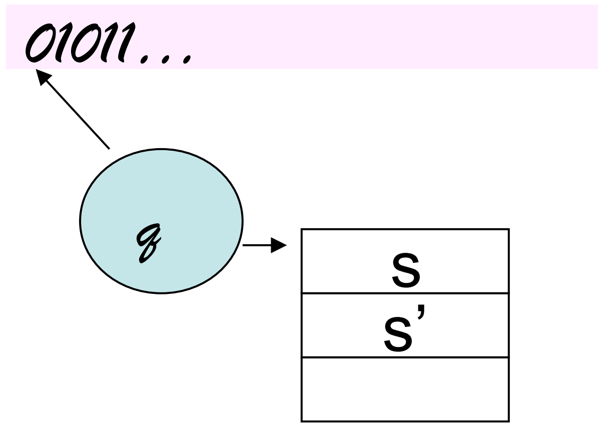
Construct G as follows:

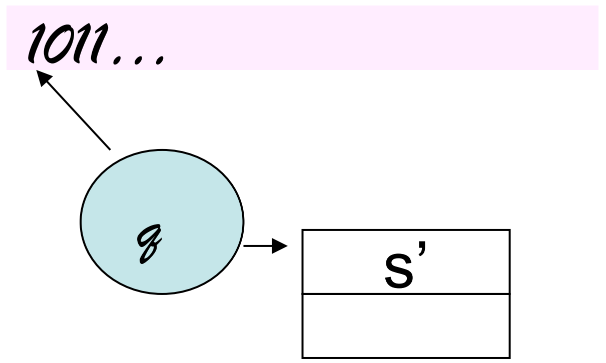
- Variables: Γ
- Terminals: Σ
- Start Symbol: s_0
- Rules: ?

Example

$f\ (q, 0, s) = \{\ (\ q, \varepsilon)\ \ldots\}$

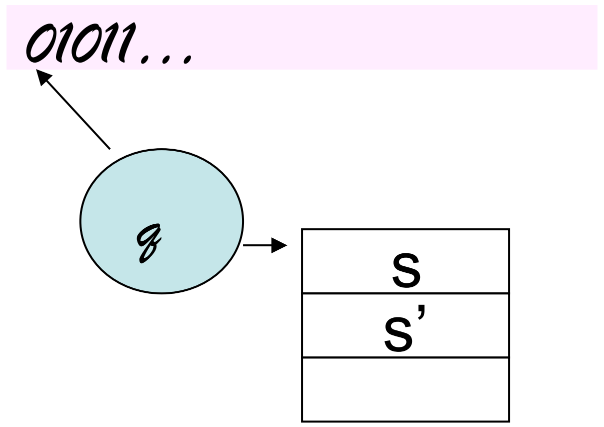
$s \rightarrow 0$

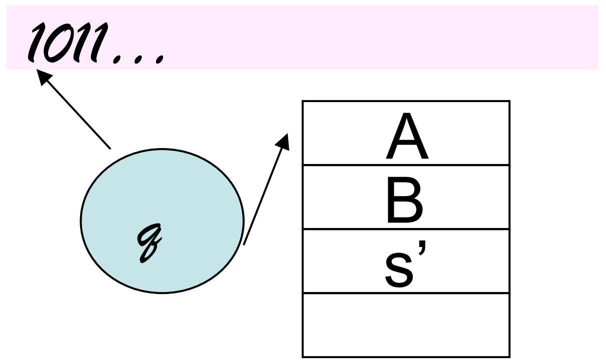




$f\ (q, 0, s) = \{\ (\ q, BA)\ \ldots\}$

$s \rightarrow 0AB$





Pushdown Automata & CFG

Theorem Let A be a pda such that $|Q| = 1$. Then $L(A)$ is a context free language (i.e., $L(A) = L(G)$ for some context free grammar).

Let $A = (\{q\}, \ \Sigma, \Gamma, f, q, s_0, \{q\})$.
Construct G as follows:

- Variables: Γ
- Terminals: Σ
- Start Symbol: s_0
- Rules: ?

$f(q, \varepsilon, s) = \{ \ (q, z), \ (q, \varepsilon), \ \ldots \}$

$s \rightarrow z^T$

$s \rightarrow \varepsilon$

$f(q, a, s) = \{ \ (q, z), \ (q, \varepsilon), \ \ldots \}$

$s \rightarrow a\, z^T$

$s \rightarrow a$