Pushdown Automata

Unit 5

Introduction

CFLs have a type of automaton that defines them

This automaton is called **Pushdown Automaton(PDA)**

It can be thought as a ε-NFA with the addition of stack

The presence of a stack means that the pushdown automata can remember infinite amount of information

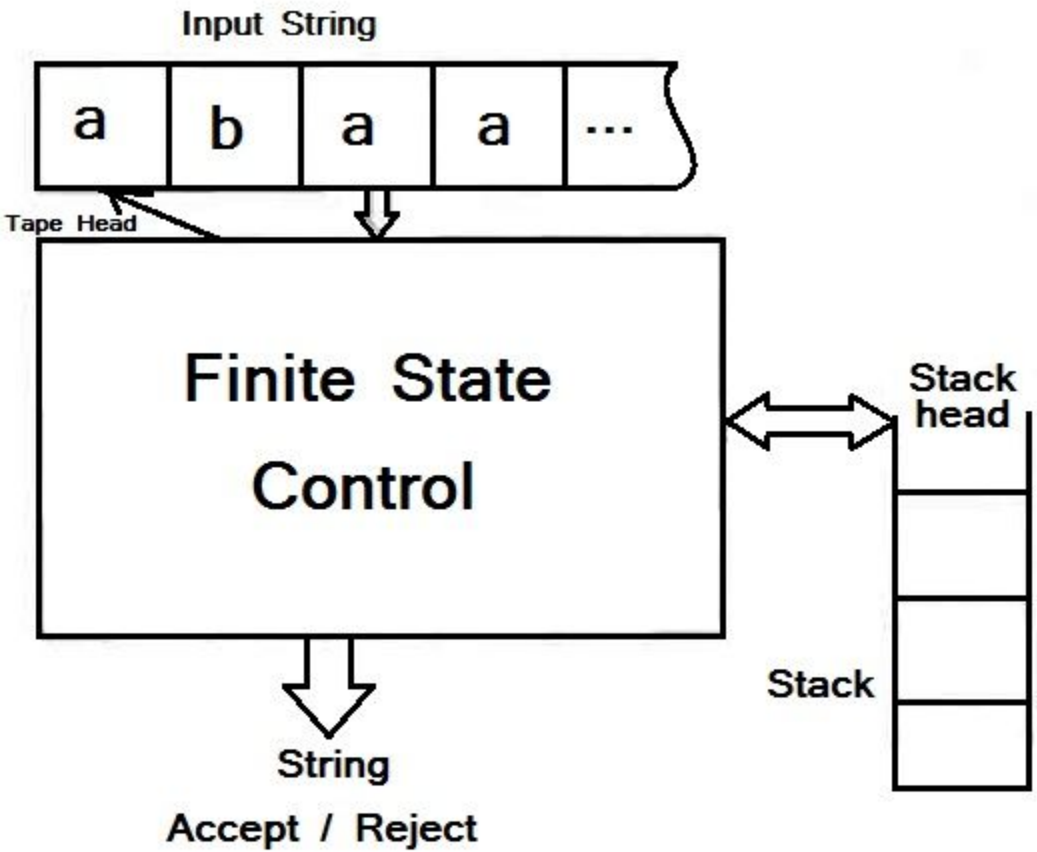
Introduction

However, pushdown automaton can only access information on its stack in a last-in-first-out way

PDA is an abstract machine determined by following three things:

(i) Input tape (ii) Finite state control (iii) A stack

We can define PDA informally as a device shown in the next figure


Input String
a
b
a
a
...
Tape Head
Finite State
Control
Stack
head
Stack
String
Accept / Reject

Move of a PDA

Each move of the machine is determined by three things:

- The current state

- Next input symbol

- Symbol on top of stack

The moves consist of:

- Changing state / staying on the same state

- Replacing the top of stack by a string of zero or more symbols

Operations of a PDA

Popping the top symbol off the stack means replacing it by ε

Pushing Y on the stack means replacing stack's top, say X, by YX, assuming the left end of stack corresponds to stack's top

A single move of the machine contains only one stack operation either push or pop

Replacing the stack symbol X by the string α can be accomplished by a sequence of basic moves (a pop followed by sequence of 0 or more pushes)

Formal Definition

A PDA P is defined by seven tuples as $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where,

Q - finite set of states

Σ - finite set of input symbols/alphabets

Γ - finite set of stack alphabets

q_0 - start state of PDA; $q_0 \in Q$

z_0 - initial stack symbol; $z_0 \in \Gamma$

F - set of final states; $F \subseteq Q$

δ - transition function that maps $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$

Formal Definition

As in finite automata, δ governs the behavior of PDA

Formally, δ takes as argument a triple (q, a, z), where,

- ‘q’ is a state

- ‘a’ is either an input symbol in Σ or ε (ε does not belong to Σ)

- ‘z’ is a stack symbol

The output of δ is a finite set of pairs (p, γ), where,

- ‘p’ is the new state

- ‘γ’ is the string on the stack after transition

Formal Definition

That is, the moves of PDA can be interpreted as:

$$\delta(q, a, z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots\ldots, (p_m, \gamma_m)\}$$

here, $q, p_i \in Q$, $a \in \Sigma \cup \varepsilon$, $z \in \Gamma$, $\gamma_i \in \Gamma^*$

It means that the PDA in state q with input symbol a and stack top z will go into state p_i and replace z by γ_i and advance to next input symbol

Graphical Representation

We can use transition diagram to represent a PDA, where

- Any state is represented by a node

- Any arc labeled with “start” (or just an arc from nowhere) to a state indicates the start state

- Doubly circled states are accepting/final states

- The arc corresponds to transition of PDA as:

an arc labeled as **a, x / α** means the transition $\delta(q, a, x) = (p, \alpha)$ for arc from state p to q

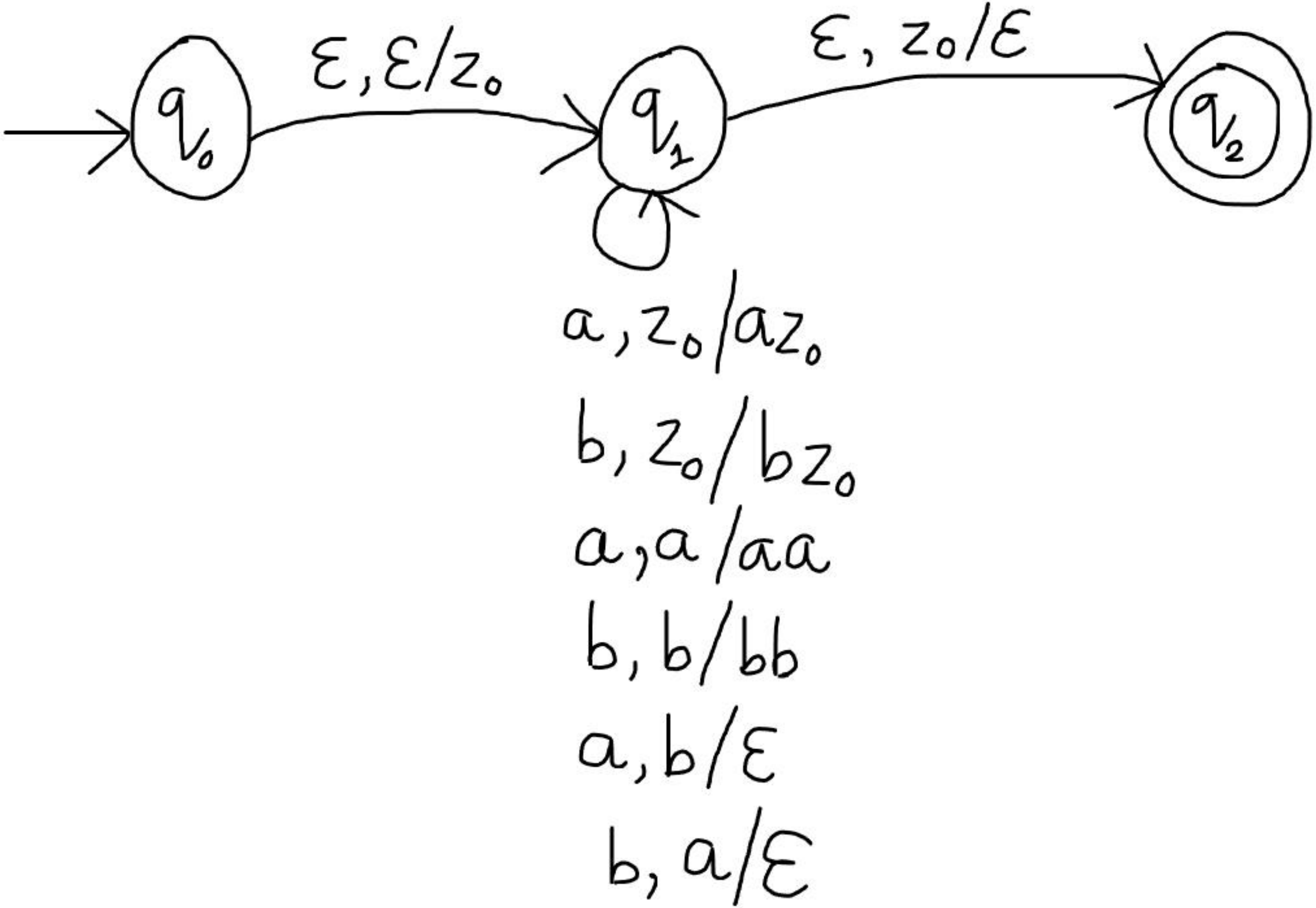
Example

(Q) Construct a PDA accepting a string over {a, b} such that number of a’s and b’s are equal

i.e. $L = \{w \mid w \in \{a, b\}^*$ and a’s and b’s are equal$\}$

(A PDA that accepts the above language can be constructed using the idea that the PDA should push the input symbol if the top of the stack symbol is same as it, otherwise pop the stack)

Transition diagram:



Example

Let P= {Q, Σ, Γ, δ, q_0, z_0, F} be the PDA recognizing the given language

here,

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, z_0\}$

$z_0 = z_0$

$q_0 = q_0$

$F = \{q_2\}$

Now, δ is defined as:

1. $\delta(q_0, \varepsilon, \varepsilon) = (q_1, z_0)$ //initialize stack to indicate the bottom symbol

2. $\delta(q_1, a, z_0) = (q_1, az_0)$

3. $\delta(q_1, b, z_0) = (q_1, bz_0)$

4. $\delta(q_1, a, a) = (q_1, aa)$

5. $\delta(q_1, b, b) = (q_1, bb)$

6. $\delta(q_1, a, b) = (q_1, \varepsilon)$

7. $\delta(q_1, b, a) = (q_1, \varepsilon)$

8. $\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$ //indicates the acceptance of string

Example

Now, tracing for string w= aabbbaab

The execution of PDA on the given string can be traced as shown next

(Practice tracing for strings “ababba” and “babab”)

S.N	State	unread string	Stack	Transition Used
1	q_0	aabbbaab	Є	--
2	q_1	aabbbaab	z_0	1
3	q_1	abbbaab	az_0	2
4	q_1	bbbaab	aaz_0	4
5	q_1	bbaab	az_0	7
6	q_1	baab	z_0	7
7	q_1	aab	bz_0	3
8	q_1	ab	z_0	6
9	q_1	b	az_0	2
10	q_1	Є	z_0	7
11	q_2	Є	Є	8

Ended up in final state q_2; hence, string is accepted

Instantaneous Description of a PDA

Any configuration of a PDA can be described by a triple (q, w, γ)

where,

q is the state.

w is the remaining input.

γ is the stack contents

Instantaneous Description of a PDA

Such description using the triple is called an instantaneous description (ID) of a PDA

For FA, the $\hat{\delta}$ notation was sufficient to represent sequences of ID through which a finite automaton moved, since the ID for a finite automaton is just its state

However, for PDAs we need a notation that describes changes in the state, the input and the stack

Instantaneous Description of a PDA

Let P = {Q, Σ, Γ, δ, q_0, z_0, F} be a PDA

Then, we define a relation ├ "yields" as

$$(q, aw, z\alpha) \vdash (p, w, \beta\alpha)$$

This move reflects the idea that, by consuming 'a' from the input, and replacing z on the top of stack by β, we can go from state q to state p

Note that what remains on the input w, and what is below the top of stack β, do not influence the action of the PDA

Instantaneous Description of a PDA

For the PDA described earlier accepting language of equal a's and b's, the accepting sequence of IDs for string "abba" can be shown as:

$$
\begin{aligned}
(q_0, abba, \varepsilon) &\vdash (q_1, abba, z_0) \\
&\vdash (q_1, bba, az_0) \\
&\vdash (q_1, ba, z_0) \\
&\vdash (q_1, a, bz_0) \\
&\vdash (q1, \varepsilon, z_0) \\
&\vdash (q_2, \varepsilon, \varepsilon) \qquad \text{Accepted}
\end{aligned}
$$

Instantaneous Description of a PDA

Therefore, $(q_0, abba, \varepsilon) \vdash^* (q_2, \varepsilon, \varepsilon)$

$\vdash$ sign is called a **turnstile notation** and represents one move

$\vdash^*$ sign represents a sequence of moves

Practice Questions

(Q1) Construct a PDA accepting language L = $\{ww^R \mid w \in (0+1)^*$ and w^R is reverse of w$\}$.

(Q2) Construct a PDA that accepts the language L = $\{a^nb^n \mid n \geq 1\}$ over {a, b}.

(Q3) Construct a PDA that accepts the language L = $\{a^nb^nc \mid n \geq 1\}$ over {a, b, c}.

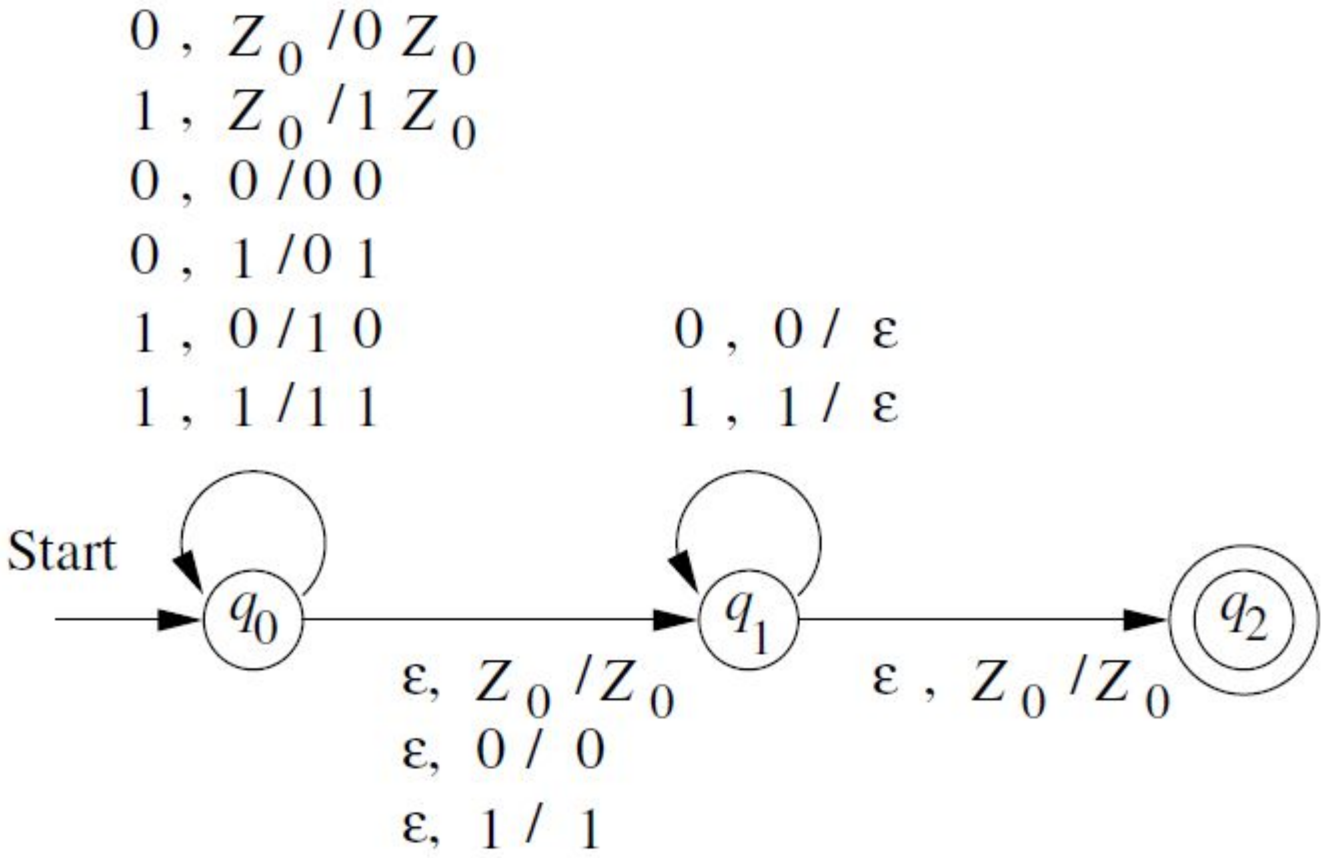
Ans 1:

(Doesn't

need to

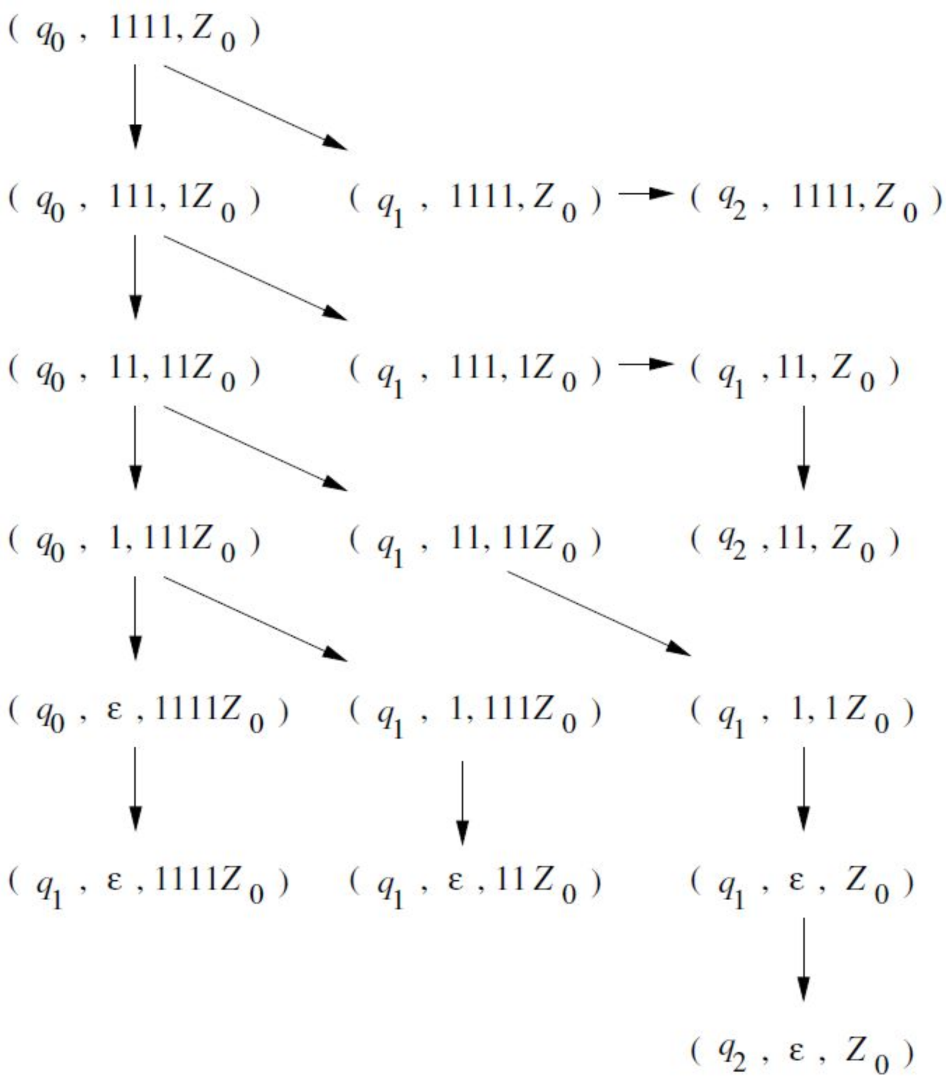
be exactly

like this)



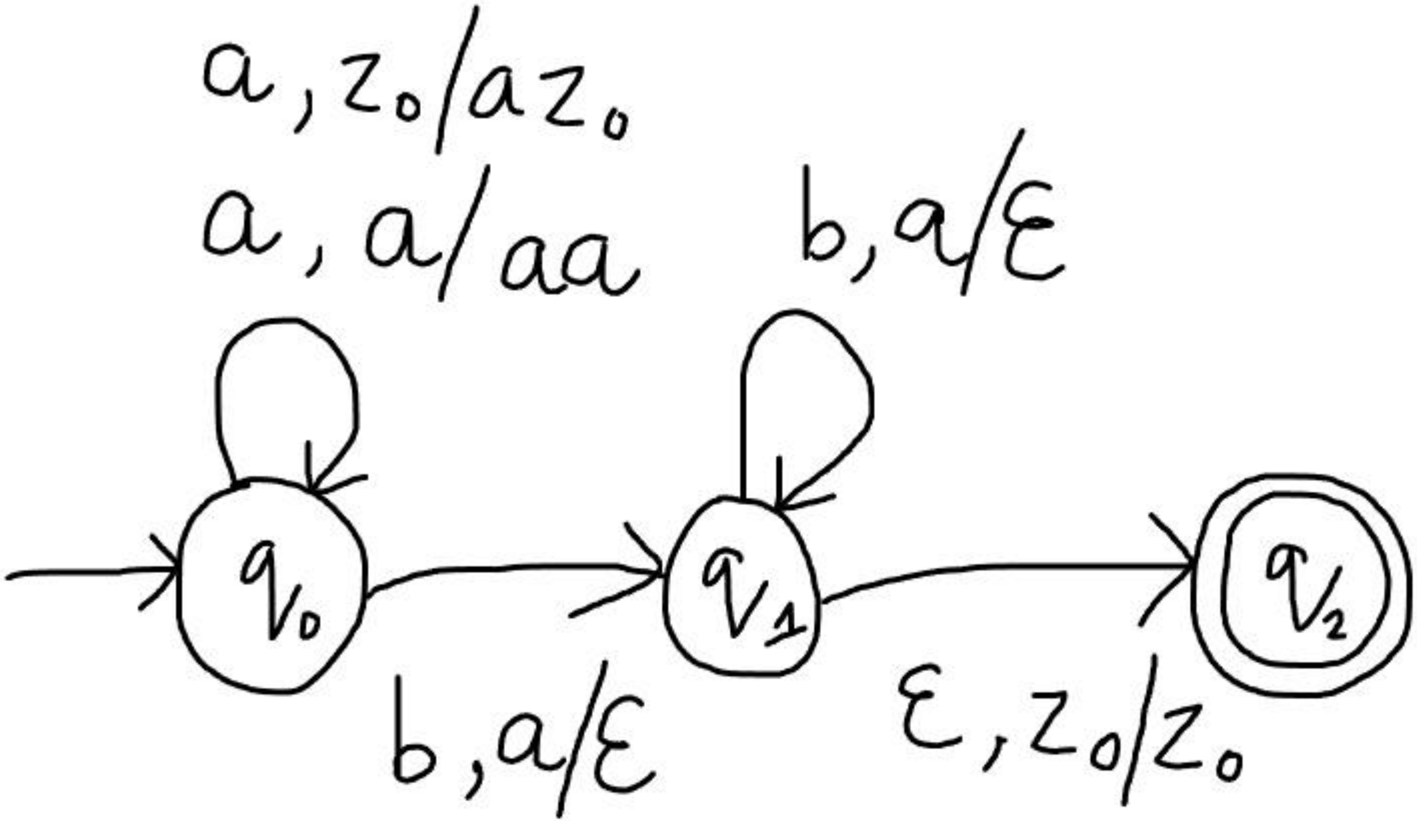
Practice Questions

The entire sequence of IDs that the PDA can reach from the initial ID (q_0, 1111, Z_0):

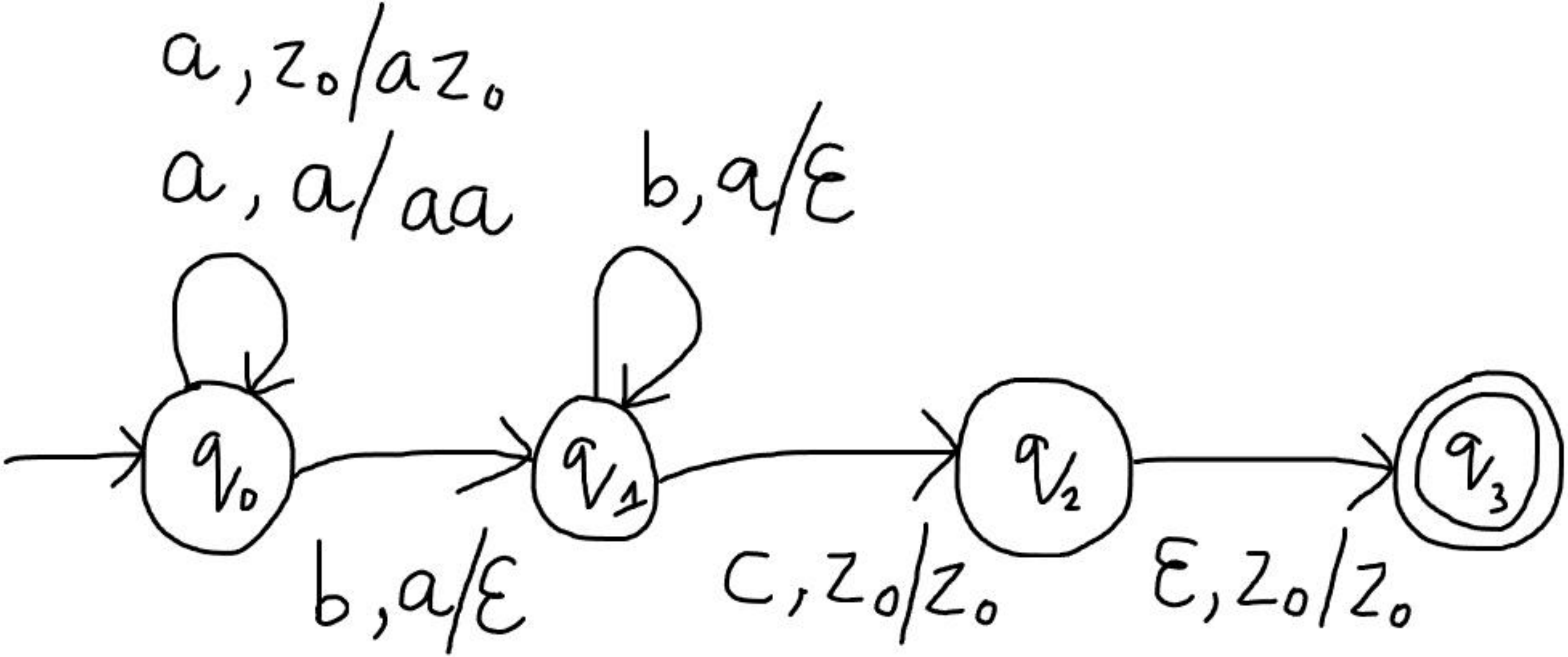


oshi

Ans 2:



Ans 3:



Deterministic Pushdown Automata (DPDA)

While the PDAs are by definition, allowed to be non-deterministic, the deterministic PDA is quite essential

In practice, the parsers generally behave like deterministic PDA, so the class of language accepted by these PDAs are practically important

DPDA are suitable for use in programming language

Deterministic Pushdown Automata

A PDA is deterministic if there is never a choice of move in any situation

These choices are of two kinds:

- If $\delta(q, a, X)$ contains more than one pair, then surely the PDA is nondeterministic because we can choose among these pairs when deciding on the next move

- However, even if $\delta(q, a, X)$ is always a singleton, we could still have a choice between using a real input symbol, or making a move on ε

Deterministic Pushdown Automata

A pushdown automata $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is deterministic if and only if the following two conditions are satisfied:

1) $\delta(q, a, X)$ has at most one member for $q \in Q$, $a \in \Sigma$ or ε and $X \in \Gamma$

2) If $\delta(q, a, X)$ is nonempty, for some $a \in \Sigma$, then $\delta(q, \varepsilon, X)$ must be empty

Example

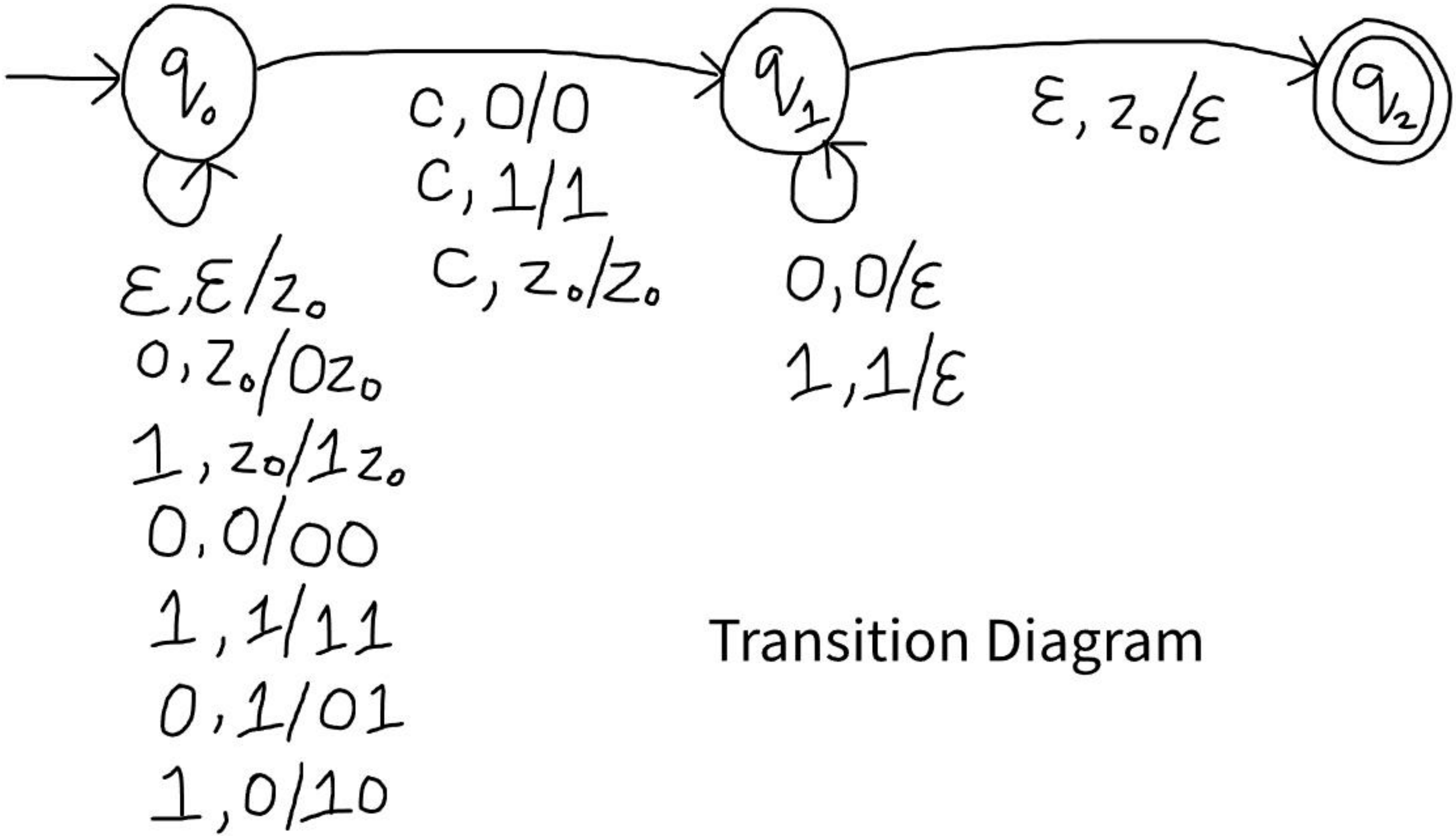
A DPDA accepting language $L = \{wcw^R \mid w \in (0+1)^*\}$ is constructed as

$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, z_0\}, \delta, q_0, z_0, \{q_2\})$, where δ is defined as:

1. $\delta(q_0, \varepsilon, \varepsilon) = (q_0, z_0)$ //initialize the stack
2. $\delta(q_0, 0, z_0) = (q_0, 0z_0)$
3. $\delta(q_0, 1, z_0) = (q_0, 1z_0)$
4. $\delta(q_0, 0, 0) = (q_0, 00)$
5. $\delta(q_0, 1, 1) = (q_0, 11)$

Example

6. $\delta(q_0, 0, 1) = (q_0, 01)$
7. $\delta(q_0, 1, 0) = (q_0, 10)$
8. $\delta(q_0, c, 0) = (q_1, 0)$ // change the state
9. $\delta(q_0, c, 1) = (q_1, 1)$ // when center
10. $\delta(q_0, c, z_0) = (q_1, z_0)$ // is reached
11. $\delta(q_1, 0, 0) = (q_1, \varepsilon)$
12. $\delta(q_1, 1, 1) = (q_1, \varepsilon)$
13. $\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$



Transition Diagram

hi

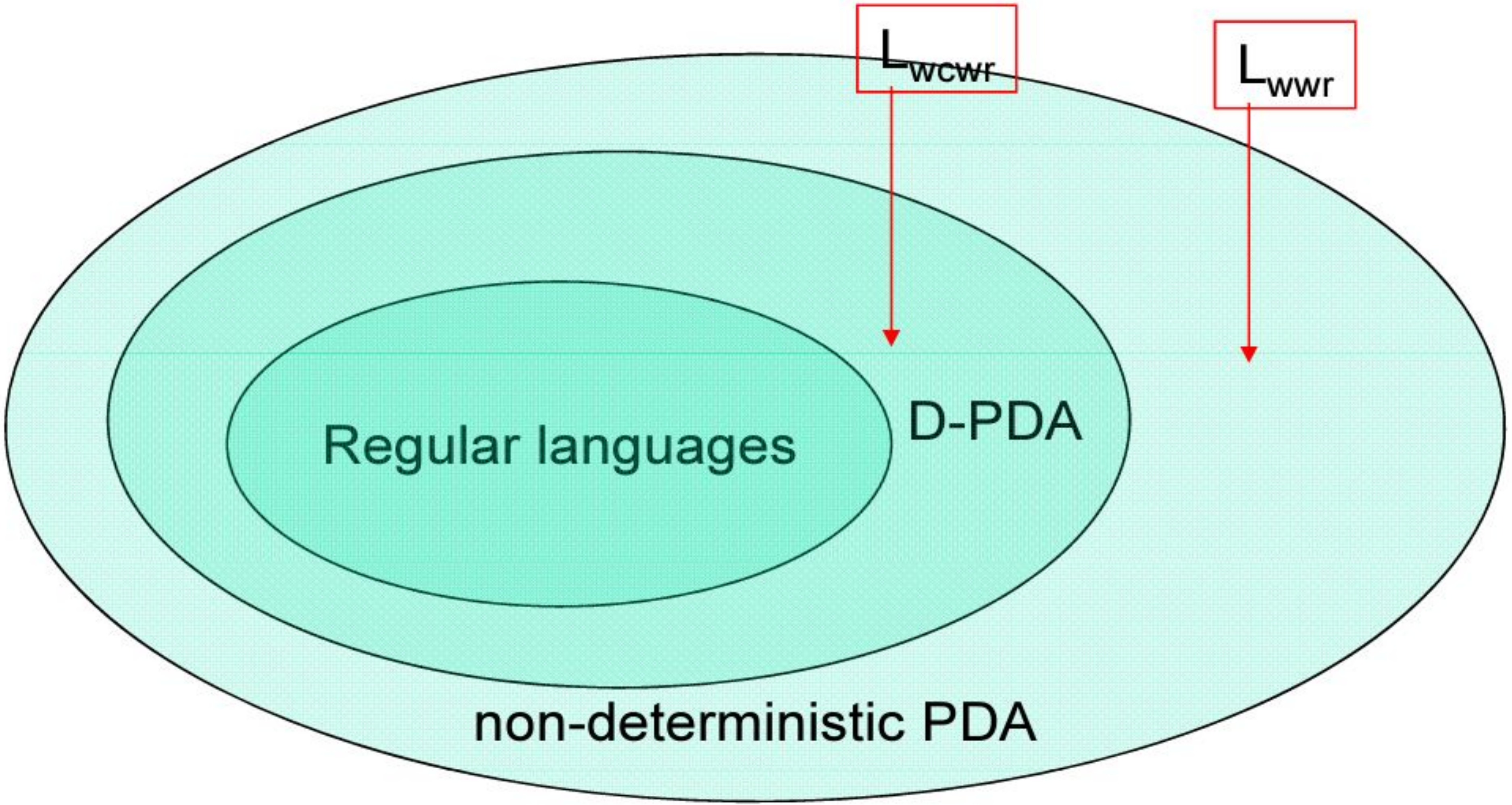
Deterministic Pushdown Automata

wcw^R - can be recognized by deterministic PDA

ww^R - needs non-deterministic PDA

NPDA is more powerful than DPDA

The language accepted by DPDA is a subset of the language accepted by NPDA



L_{wcwr}
L_{wwr}
D-PDA
Regular languages
non-deterministic PDA

Example

(Q) Construct a DPDA that accepts strings of balanced ordered parenthesis.

<u>Ans:</u> Required DPDA can be defined as

$Q = (q_0, q_1, q_2)$, $\Sigma = \{ \{, \}, (,), [,] \}$, $\Gamma = \Sigma \cup \{z_0\}$, δ, q_0, z_0, $F = \{q_2\}$

here, δ is defined as follows:

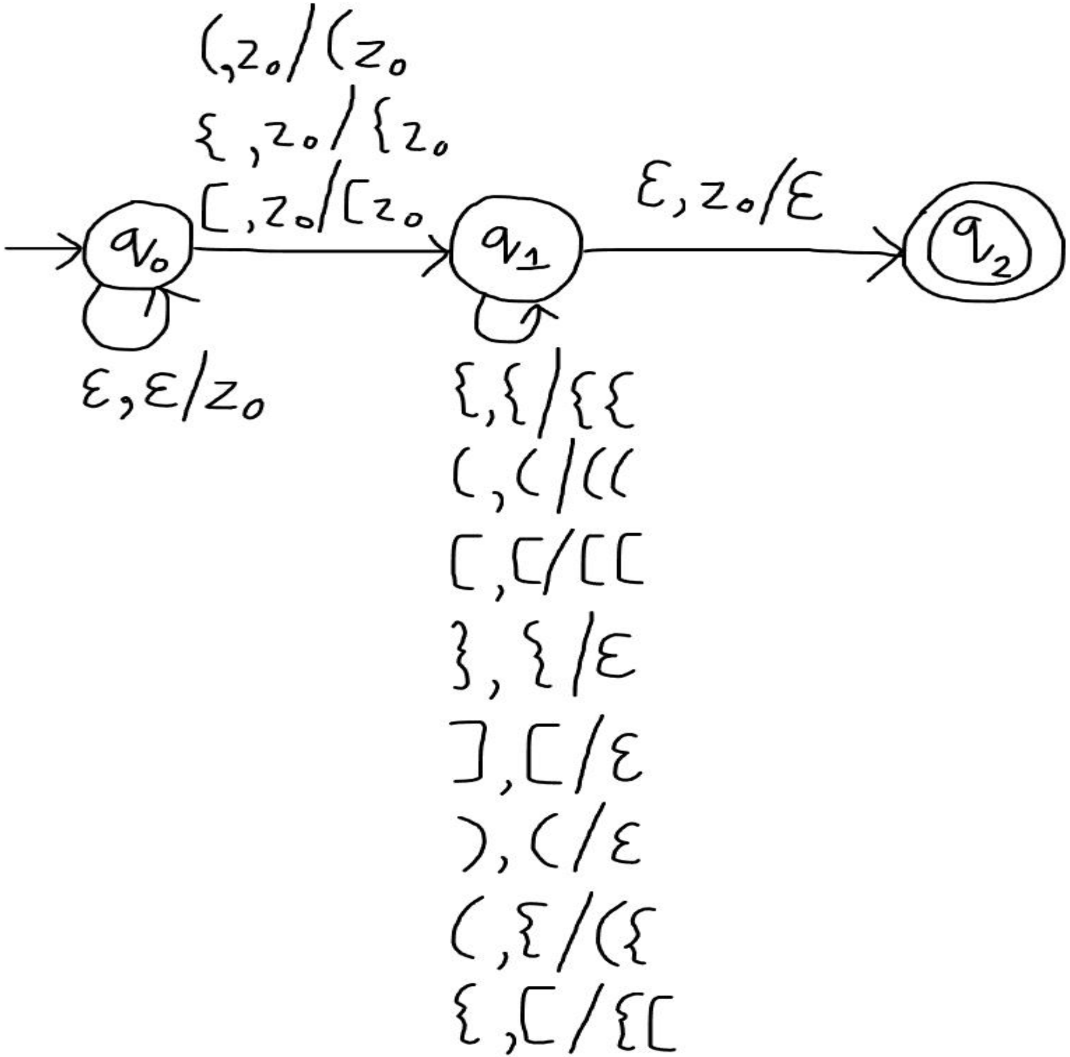
Example

1. $\delta(q_0, \varepsilon, \varepsilon) = (q_0, z_0)$
2. $\delta(q_0, \{, z_0) = (q_1, \{ z_0)$
3. $\delta(q_0, [, z_0) = (q_1, [z_0)$
4. $\delta(q_0, (, z_0) = (q_1, (z_0)$
5. $\delta(q_1, \{, \{) = (q_1, \{ \{)$
6. $\delta(q_1, [, [ ) = (q_1, [[ )$
7. $\delta(q_1, (, ( ) = (q_1, (( )$
8. $\delta(q_1, \}, \{) = (q_1, \varepsilon)$
9. $\delta(q_1,], [ ) = (q_1, \varepsilon)$
10. $\delta(q_1,), ( ) = (q_1, \varepsilon)$
11. $\delta(q_1, (, \{) = (q_1, (\{)$
12. $\delta(q_1, \{, [ ) = (q_1, \{ [ )$
13. $\delta(q_1, \varepsilon, z_0) = (q_2, \varepsilon)$

Example

Transition

diagram:



Example

Tracing for the string "[{ }]"

$(q_0, [\,\{\,\}\,], z_0) \vdash (q_1, \{\,\}\,], [\,z_0)$

$\vdash (q_1, \}\,], \{\,[\,z_0)$

$\vdash (q_1,], [\,z_0)$

$\vdash (q_1, \varepsilon, z_0)$

$\vdash (q_2, \varepsilon, \varepsilon)$ Accepted

Language of a PDA

We can define acceptance of any string by a PDA in two ways:

(1) Acceptance by final state

- Given a PDA P, the language accepted by P by final state, L(P), is:

$\{w \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \gamma)\}$ where $q \in F$ and $\gamma \in \Gamma^*$

Language of a PDA

(2) Acceptance by empty stack

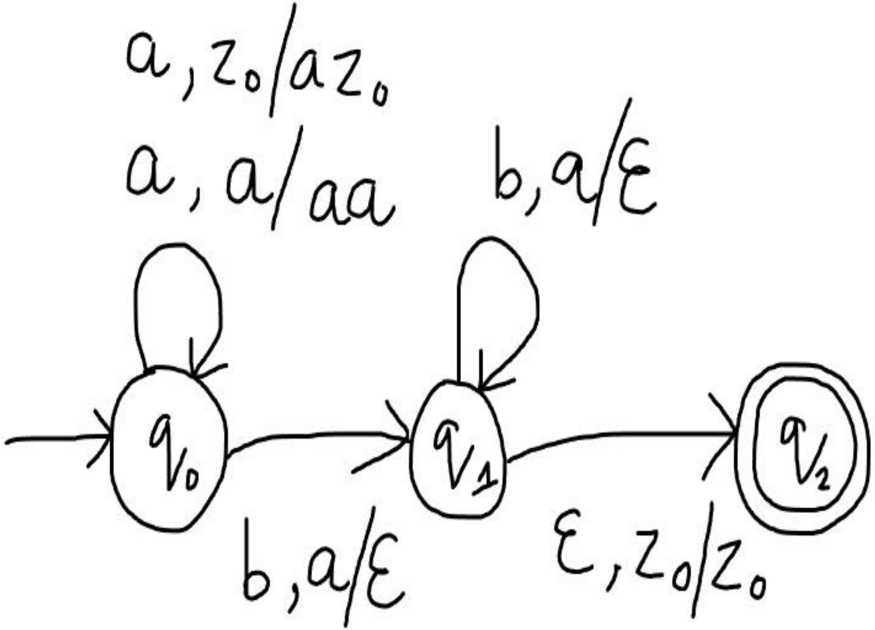
- Given a PDA P, the language accepted by P by empty stack, L(P), is:

$\{w \mid (q_0, w, z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$ where $q \in Q$

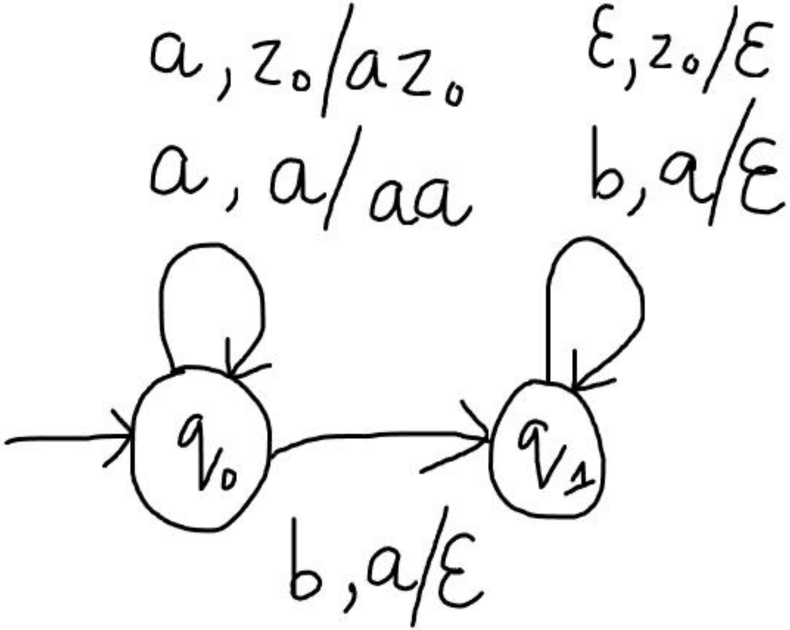
Example

L = $\{a^n b^n \mid n \geq 1\}$

Acceptance by final state



Acceptance by empty stack



Acceptance of Strings by PDA

PDAs accepting by final state and PDAs accepting by empty stack are **equivalent**

$P_F \rightarrow$ PDA accepting by final state

$$P_F = (Q_F, \Sigma, \Gamma, \delta_F, q_0, \Gamma, z_0, F)$$

$P_N \rightarrow$ PDA accepting by empty stack

$$P_N = (Q_N, \Sigma, \Gamma, \delta_N, q_0, z_0)$$

Acceptance of Strings by PDA

<u>Theorem:</u>

- ($P_N \Rightarrow P_F$): For every P_N, there exists a P_F s.t. $L(P_F) = L(P_N)$

- ($P_F \Rightarrow P_N$): For every P_F, there exists a P_N s.t. $L(P_F) = L(P_N)$

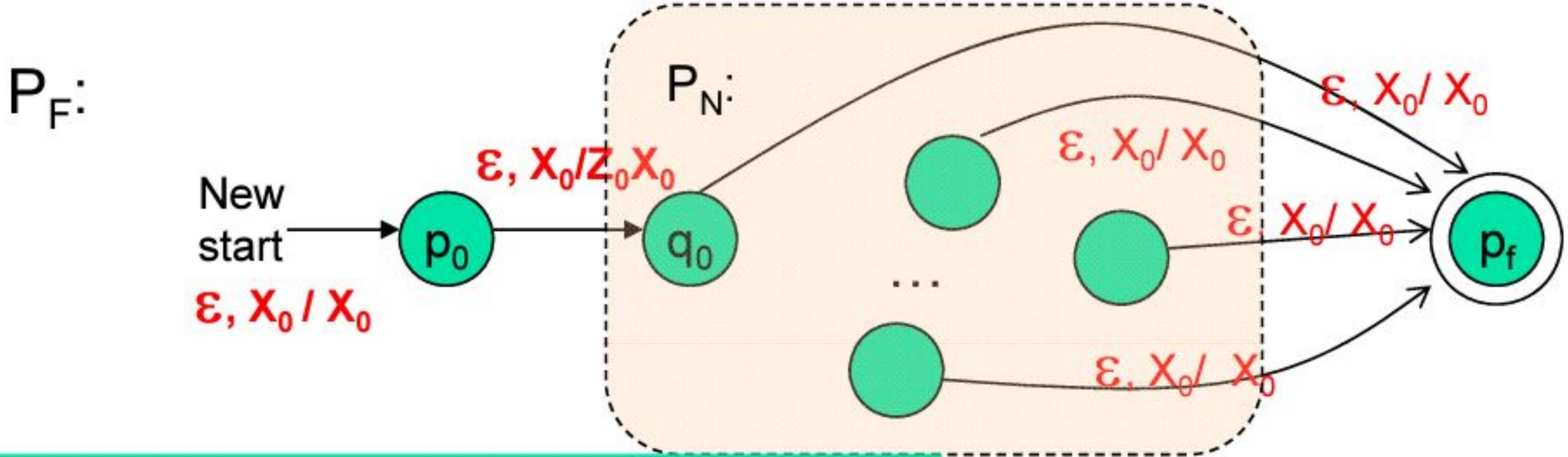
Conversion of Empty Stack PDA to Final State PDA

Whenever P_N's stack becomes empty, make P_F go to a final state without consuming any addition symbol

To detect empty stack in P_N:

P_F initially pushes a new stack symbol x_0 (not in Γ of P_N) before simulating P_N

Conversion of Empty Stack PDA to Final State PDA



$$P_F = (Q_N \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

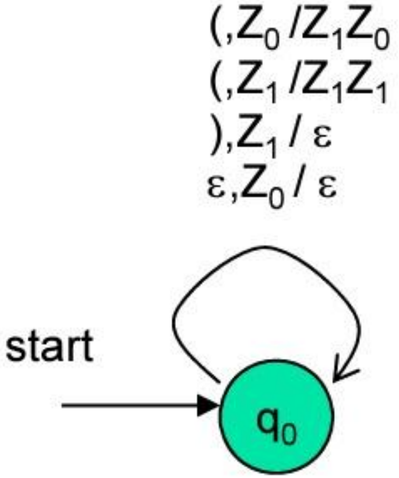
Example

Given PDA accepting by empty stack for matching parenthesis “(“ and “)”. Convert this PDA into one accepting by final state.

P_N: $(\{q_0\}, \{(,)\}, \{Z_0, Z_1\}, \delta_N, q_0, Z_0)$

δ_N:

$$\delta_N(q_0, (, Z_0) = \{ (q_0, Z_1Z_0) \}$$
$$\delta_N(q_0, (, Z_1) = \{ (q_0, Z_1Z_1) \}$$
$$\delta_N(q_0,), Z_1) = \{ (q_0, \varepsilon) \}$$
$$\delta_N(q_0, \varepsilon, Z_0) = \{ (q_0, \varepsilon) \}$$



P_f: $(\{p_0, q_0, p_f\}, \{(,)\}, \{X_0, Z_0, Z_1\}, \delta_f, p_0, X_0, p_f)$

Example

Converted PDA:

δ_f:

$\delta_f(p_0, \varepsilon, X_0) = \{ (q_0, Z_0) \}$

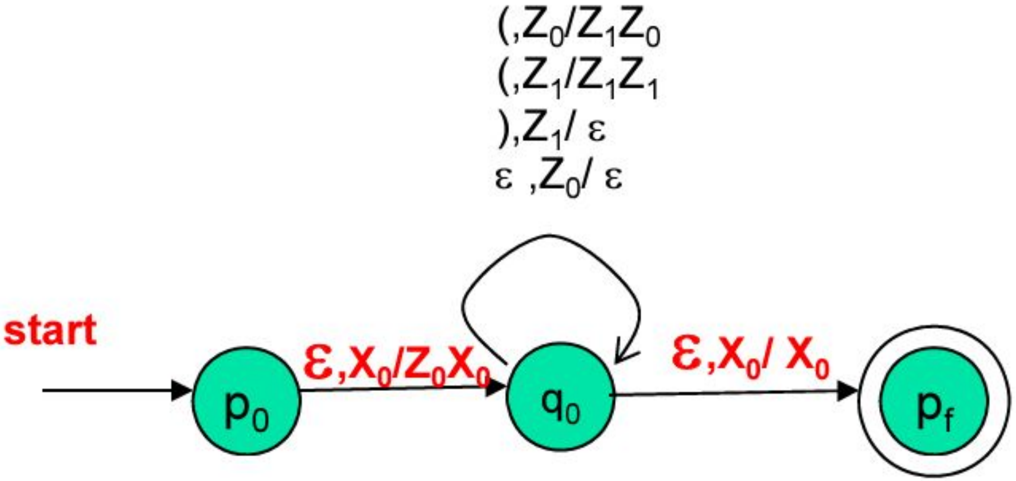
$\delta_f(q_0, (, Z_0) = \{ (q_0, Z_1 Z_0) \}$

$\delta_f(q_0, (, Z_1) = \{ (q_0, Z_1 Z_1) \}$

$\delta_f(q_0,), Z_1) = \{ (q_0, \varepsilon) \}$

$\delta_f(q_0, \varepsilon, Z_0) = \{ (q_0, \varepsilon) \}$

$\delta_f(p_0, \varepsilon, X_0) = \{ (p_f, X_0) \}$



Conversion of Final State PDA to Empty Stack PDA

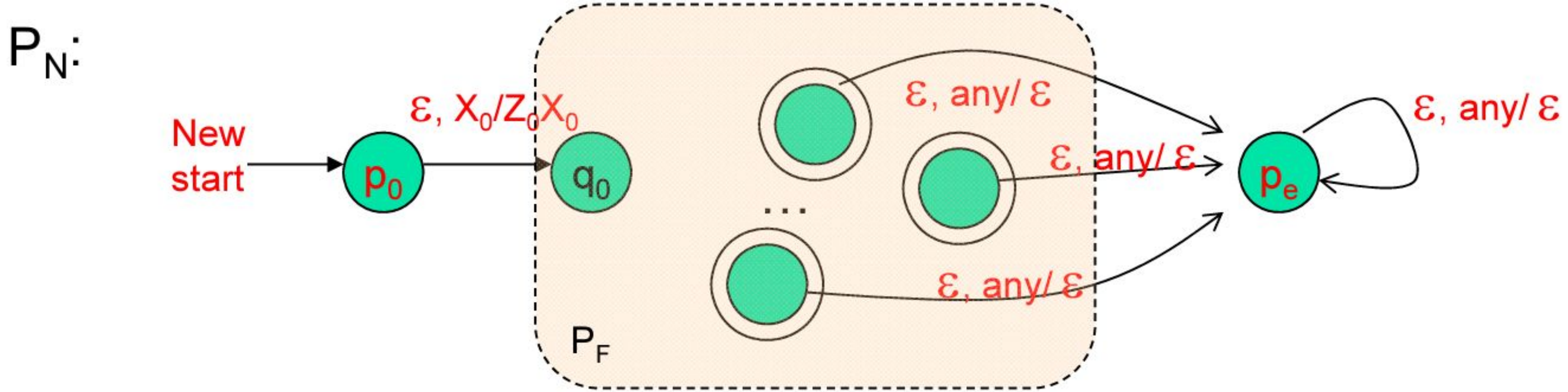
Whenever P_F reaches a final state, just make an ε-transition into a new end state, clear out the stack and accept

Caution: What if P_F design is such that it clears the stack midway without entering a final state?

To address this, add a new start symbol x_0 (not in Γ of P_F)

Conversion of Final State PDA to Empty Stack PDA

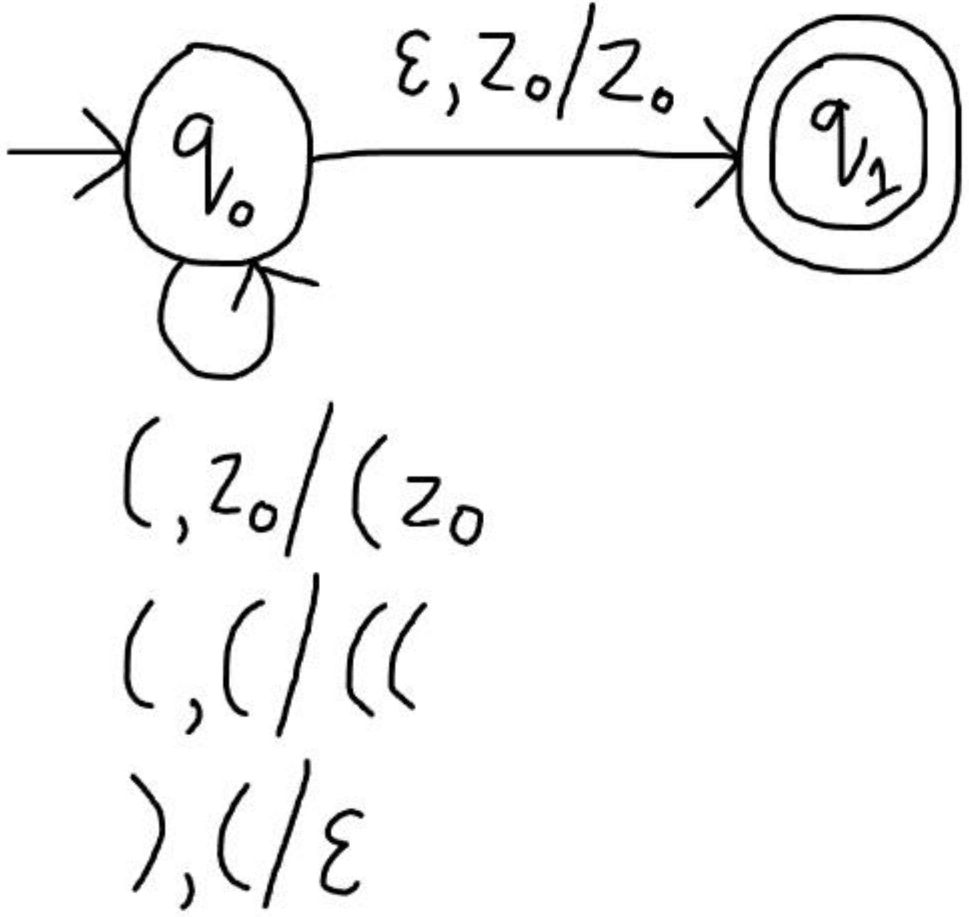
$P_N = (Q \cup \{p_0, p_e\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$



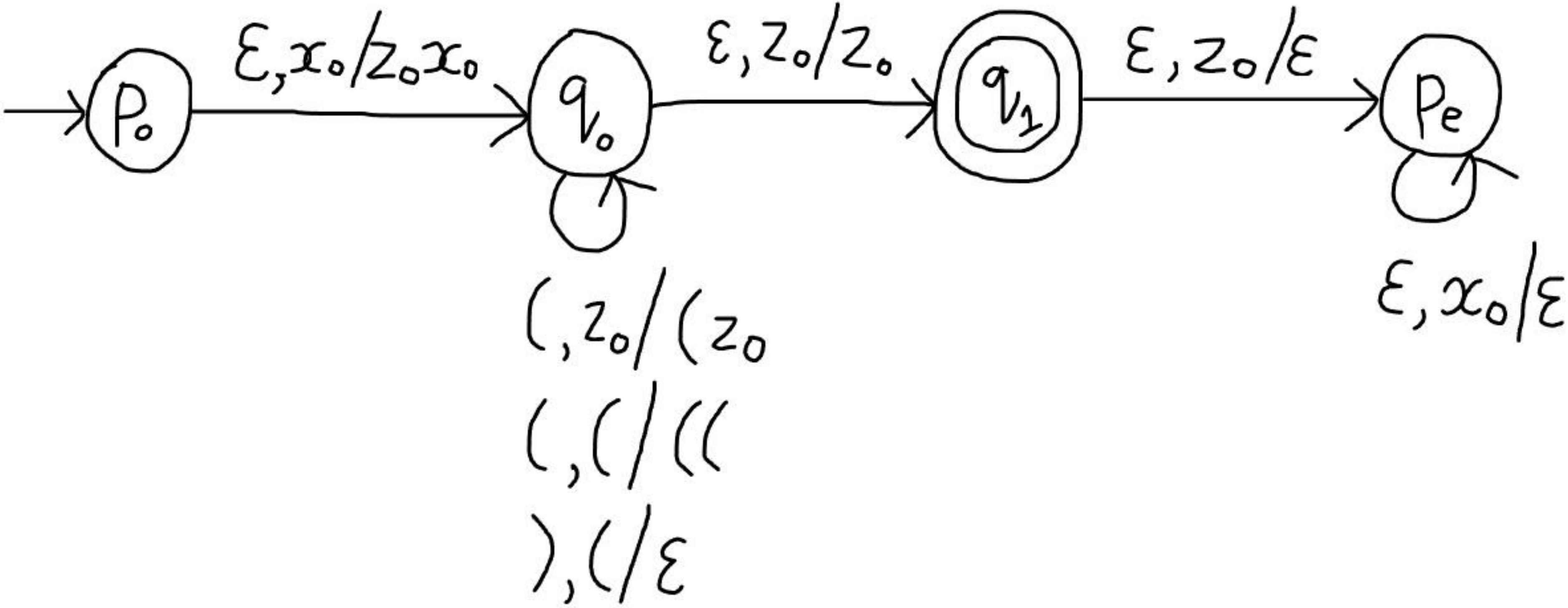
Example

Given PDA accepting by final state for matching parenthesis "(" and ")".

Convert this PDA into one accepting by empty stack.



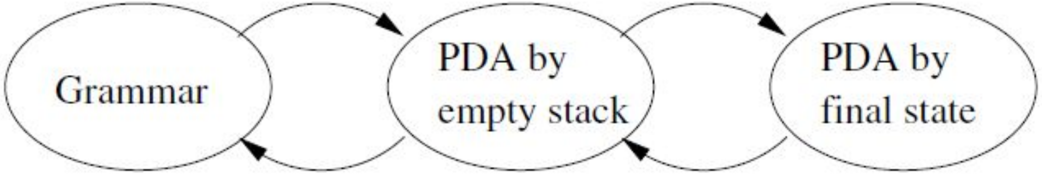
Converted PDA:



Equivalence of PDAs and CFGs

The languages defined by PDAs are exactly the context free languages

We can show that the languages defined by CFGs and the languages accepted by PDAs are **equivalent**



Conversion of CFG to PDA

Given a CFG G =(V, T, P, S), we can construct a pushdown automaton, M which accepts the language generated by the grammar G, i.e. L(M) = L (G)

The machine can be defined as:

$$M = (\{q\}, T, V \cup T, \delta, q, S, \emptyset)$$

where,

Q = {q} is only the state in the PDA

Σ = T

Conversion of CFG to PDA

$\Gamma = V \cup T$ (i.e. PDA uses terminals & variables of G as stack symbols)

$z_0 = S$ (initial stack symbol is the start symbol in grammar)

$F = \emptyset$

$q_0 = q$

and, δ can be defined as: [all transitions occur on a single state]

(i) $\delta(q, \varepsilon, A) = (q, \alpha)$ where $A \rightarrow \alpha$ is a production of G

(ii) $\delta(q, a, a) = (q, \varepsilon)$ for all $a \in T$

Conversion of CFG to PDA

Alternatively, we can define PDA for the CFG with two states p & q, with p being the start state

Here, the idea is that the stack symbol is initially supposed to be ε, and the PDA first starts in state p reading ε

It inserts start symbol 'S' of CFG into stack and changes state to q

Then all transitions occur in state q

Conversion of CFG to PDA

That is, the PDA can be defined as:

$$M = (\{p, q\}, T, V \cup T, \delta, p, \{q\}) \quad \text{with stack top} = \varepsilon$$

Then, δ can be defined as:

(i) $\delta(p, \varepsilon, \varepsilon) = (q, S)$ where S is the start symbol of grammar G

(ii) $\delta(q, \varepsilon, A) = (q, \alpha)$ where $A \rightarrow \alpha$ is a production of G

(iii) $\delta(q, a, a) = (q, \varepsilon)$ for all $a \in T$

Example

Let G = (V, T, P, S) where P is

$$E \rightarrow T \mid E + T$$

$$T \rightarrow F \mid T * F$$

$$F \rightarrow a \mid (E)$$

PDA equivalent to G can be defined as :

$$M = (\{q_0\}, \{ a, *, +, (,) \}, \{ a, *, +, (,), E, T, F \}, \delta, q_0, E, \emptyset\}$$

Example

where δ can be defined as:

$\delta(q_0, \varepsilon, E) = \{ (q_0, T), (q_0, E + T) \}$

$\delta(q_0, \varepsilon, F) = \{ (q_0, a), (q_0, (E)) \}$

$\delta(q_0, *, *) = \{ (q_0, \varepsilon) \}$

$\delta(q_0, (, () = \{ (q_0, \varepsilon) \}$

$\delta(q_0, \varepsilon, T) = \{ (q_0, F), (q_0, T * F) \}$

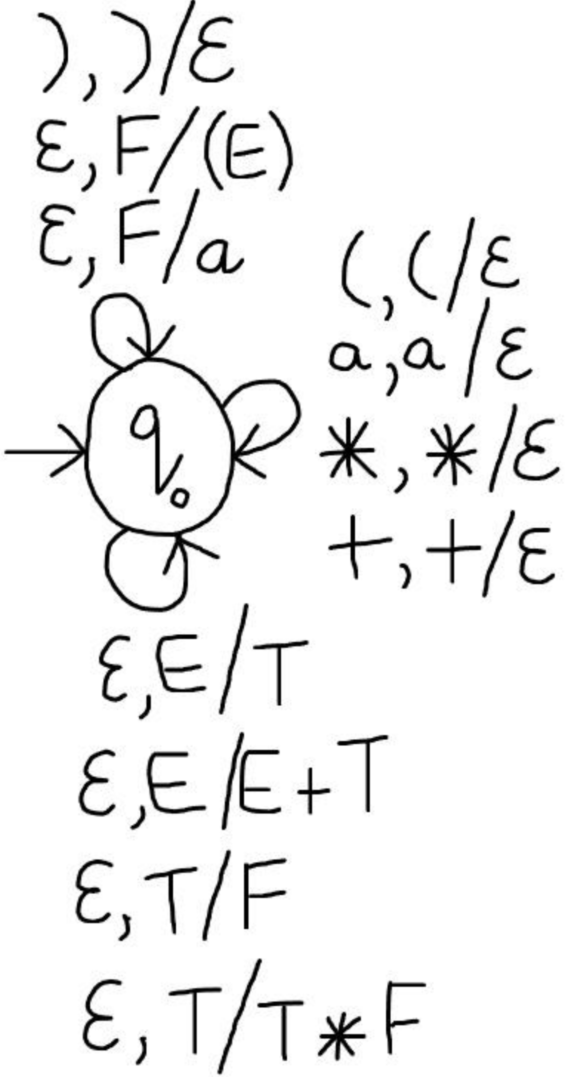
$\delta(q_0, a, a) = \{ (q_0, \varepsilon) \}$

$\delta(q_0, +, +) = \{ (q_0, \varepsilon) \}$

$\delta(q_0,),)) = \{ (q_0, \varepsilon) \}$

Example

Transition diagram:



Tracing the acceptance of a + (a * a), starting from (q_0, a + (a * a), E)

$\vdash$ (q0, a + (a*a), E + T)

$\vdash$ (q0, a + (a*a), T + T)

$\vdash$ (q0, a + (a*a), F + T)

$\vdash$ (q0, a + (a*a), a + T)

$\vdash$ (q0, + (a*a), + T)

$\vdash$ (q0, (a*a), T)

$\vdash$ (q0, (a*a), F)

$\vdash$ (q0, (a*a), (E))

$\vdash$ (q0, a*a), E))

$\vdash$ (q0, a*a), T))

$\vdash$ (q0, a*a), T*F)

$\vdash$ (q0, a*a), F*F)

$\vdash$ (q0, a*a), a*F)

$\vdash$ (q0, *a), *F)

$\vdash$ (q0, a), F))

$\vdash$ (q0, a), a))

$\vdash$ (q0,),))

$\vdash$ (q0, Є, Є) Accepted

Example

Tracing using CFG:

$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + F \Rightarrow a + (E) \Rightarrow a + (T * F)$

$\Rightarrow a + (F * F) \Rightarrow a + (a * F) \Rightarrow a + (a * a)$

Practice Question

(Q) Given grammar G = (V, T, P, S) with following productions:

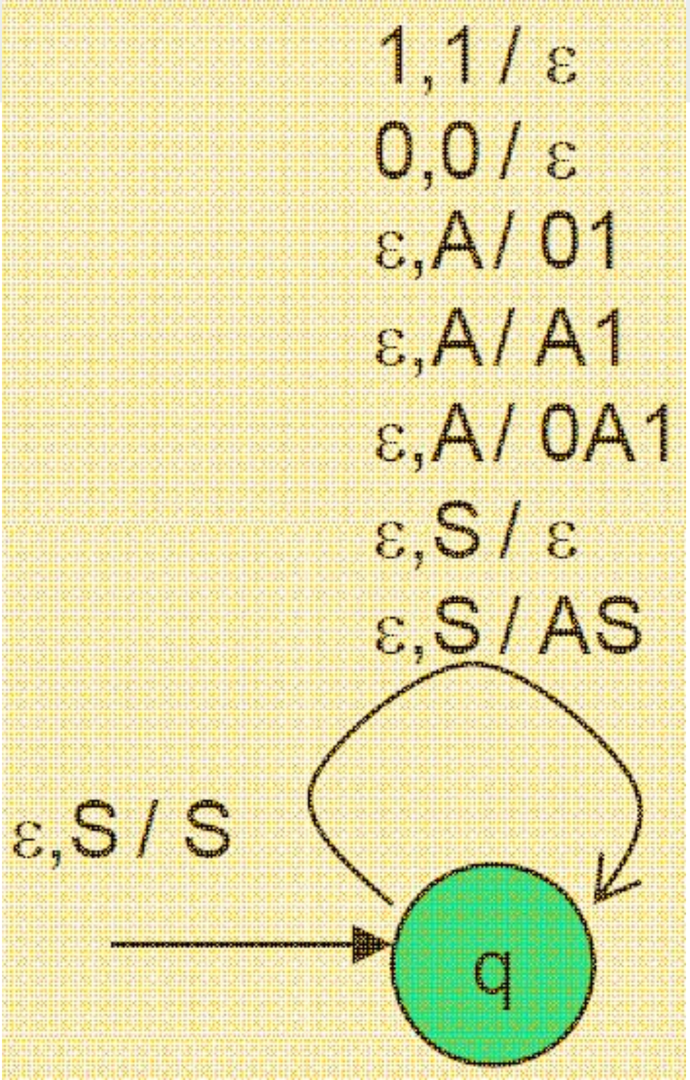
$$S \rightarrow AS \mid \varepsilon$$

$$A \rightarrow 0A1 \mid A1 \mid 01$$

Convert it into an equivalent PDA.

Practice Question

Ans:



Conversion of PDA to CFG

Given a PDA M = (Q, Σ, Γ, δ, q_0, z_0, F); F = Ø, we can obtain an equivalent CFG, G = (V, T, P, S) which generates the same language as accepted by the PDA M

The set of variables V in the grammar consist of:

- The special symbol S, which is start symbol.

- All the symbols of the form [p x q]; for all p, q ∈ Q and X ∈ Γ

i.e. V = {S} ∪ { [p x q] }

The terminal in the grammar, T = Σ

Conversion of PDA to CFG

The productions of G is as follows:

- For all states $q \in Q$, $S \rightarrow [q_0\ z_0\ q]$ is a production of G

- For any state $p, q \in Q$, $x \in \Gamma$ and $a \in \Sigma \cup \{\varepsilon\}$, if $\delta(q, a, x) = (p, \varepsilon)$ then

$[q\ x\ p] \rightarrow a$

- For any state $p, q \in Q$, $x \in \Gamma$ and $a \in \Sigma \cup \{\varepsilon\}$, if $\delta(q, a, x) = (p, Y_1, Y_2, \ldots, Y_k)$; where $Y_1, Y_2, \ldots, Y_k \in \Gamma$ and $k \geq 0$, then for all lists of states $P_1, P_2, \ldots, P_k$, G has the production $[p\ x\ q] \rightarrow a\ [p\ Y_1 P_1]\ [P_1 Y_2 P_2] \ldots [P_{k-1} Y_k P_k]$

Conversion of PDA to CFG

The last production says that one way to pop x and go from state q to state P_k is to read "a" (which may be ε), then use some input to pop Y_1 off the stack while going from state p to P_1, then read some more input that pops Y_2 off the stack and goes from P_1 to P_2 and so on

Example

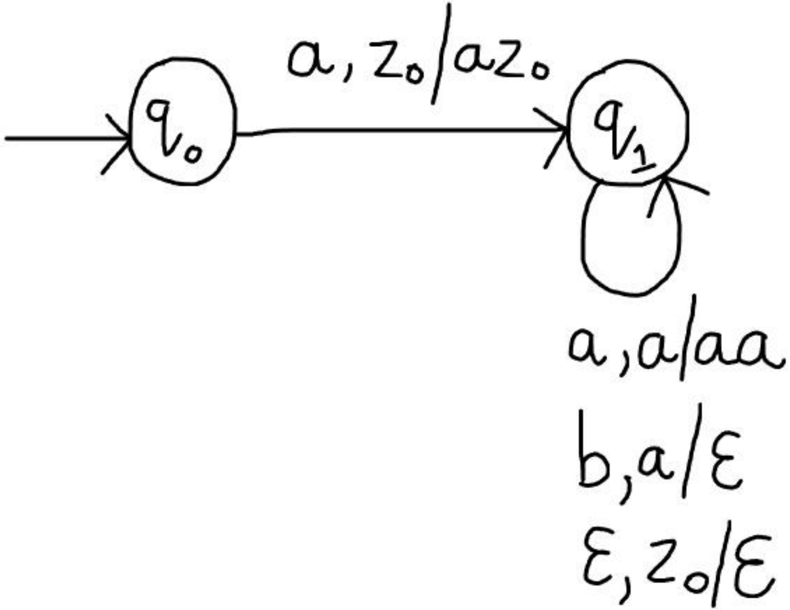
Let a PDA that recognizes the language $L = \{a^n b^n \mid n > 0\}$ be defined as:

(1) $\delta(q_0, a, z_0) = (q_1, az_0)$

(2) $\delta(q_1, a, a) = (q_1, aa)$

(3) $\delta(q_1, b, a) = (q_1, \varepsilon)$

(4) $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$



Example

Let G = (V, T, P, S) be the equivalent CFG for given PDA where,

V = {S} ∪ { [p x q] } p, q ∈ Q, x ∈ Γ

i.e. V = {S, $[q_0\, z_0\, q_0]$, $[q_0\, z_0\, q_1]$, $[q_1\, z_0\, q_0]$, $[q_1\, z_0\, q_1]$, $[q_0\, a\, q_0]$, $[q_0\, a\, q_1]$,

$[q_1\, a\, q_0]$, $[q_1\, a\, q_1]$, $[q_0\, b\, q_0]$, $[q_0\, b\, q_1]$, $[q_1\, b\, q_0]$, $[q_1\, b\, q_1]$ }

T= Σ = {a, b}

S = S

Example

P is defined by the following productions:

(1) $S \rightarrow [q_0\, z_0\, q_0] \mid [q_0\, z_0\, q_1]$ i.e. $S \rightarrow [q_0\, z_0\, r_2]$ for r_2 in $\{q_0, q_1\}$

(2) For $\delta(q_0, a, z_0) = (q_1, az_0)$,

$[q_0\, z_0\, q_0] \rightarrow a\, [q_1\, a\, q_0]\, [q_0\, z_0\, q_0]$

$[q_0\, z_0\, q_0] \rightarrow a\, [q_1\, a\, q_1]\, [q_1\, z_0\, q_0]$

$[q_0\, z_0\, q_1] \rightarrow a\, [q_1\, a\, q_0]\, [q_0\, z_0\, q_1]$

$[q_0\, z_0\, q_1] \rightarrow a\, [q_1\, a\, q_1]\, [q_1\, z_0\, q_1]$

These productions can be simplified as:

$[q_0\, z_0\, r_2] \rightarrow a\, [q_1\, a\, r_1]\, [r_1\, z_0\, r_2]$

for r_i in $\{q_0, q_1\}$

Example

(3) For $\delta(q_1, a, a) = (q_1, aa)$,

$[q_1\ a\ q_0] \rightarrow a\ [q_1\ a\ q_0]\ [q_0\ a\ q_0]$

$[q_1\ a\ q_0] \rightarrow a\ [q_1\ a\ q_1]\ [q_1\ a\ q_0]$

$[q_1\ a\ q_1] \rightarrow a\ [q_1\ a\ q_0]\ [q_0\ a\ q_1]$

$[q_1\ a\ q_1] \rightarrow a\ [q_1\ a\ q_1]\ [q_1\ a\ q_1]$

These productions can be simplified as:

$[q_1\ a\ r_2] \rightarrow a\ [q_1\ a\ r_1]\ [r_1\ a\ r_2]$

for r_i in $\{q_0, q_1\}$

Example

(4) For $\delta(q_1, b, a) = (q_1, \varepsilon)$,

$[q_1\ a\ q_1] \rightarrow b$

(5) For $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$,

$[q_1\ z_0\ q_1] \rightarrow \varepsilon$

Summary of productions:

$S \rightarrow [q_0\ z_0\ r_2]$

$[q_0\ z_0\ r_2] \rightarrow a\ [q_1\ a\ r_1]\ [r_1\ z_0\ r_2]$

$[q_1\ a\ r_2] \rightarrow a\ [q_1\ a\ r_1]\ [r_1\ a\ r_2]$

$[q_1\ a\ q_1] \rightarrow b$

$[q_1\ z_0\ q_1] \rightarrow \varepsilon$

Example

Derivation of string “aabb” can be shown as:

$$S \Rightarrow [q_0\, z_0\, r_2]$$

$$\Rightarrow a\,[q_1\, a\, r_1]\,[r_1\, z_0\, r_2]$$

$$\Rightarrow aa\,[q_1\, a\, r_1]\,[r_1\, a\, r_2]\,[r_1\, z_0\, r_2]$$

$$\Rightarrow aab\,[r_1\, a\, r_2]\,[r_1\, z_0\, r_2]$$

$$\Rightarrow aabb\,[r_1\, z_0\, r_2] \Rightarrow aabb\varepsilon \Rightarrow aabb$$

Practice Question

(Q) Given the following PDA P for bracket matching of "(" and ")":

$P = (\{q_0\}, \{b, e\}, \{Z_0, Z_1\}, \delta, q_0, Z_0)$ here, b = "(" and e = ")"

(1) $\delta(q_0, b, Z_0) = (q_0, Z_1Z_0)$

(2) $\delta(q_0, b, Z_1) = (q_0, Z_1Z_1)$

(3) $\delta(q_0, e, Z_1) = (q_0, \varepsilon)$

(4) $\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon)$

Convert it into equivalent CFG.

Practice Question

Ans: The productions of the resultant CFG will be as follows:

$$S \rightarrow [q_0 Z_0 q_0]$$

$$[q_0 Z_0 q_0] \rightarrow b [q_0 Z_1 q_0] [q_0 Z_0 q_0]$$

$$[q_0 Z_1 q_0] \rightarrow b [q_0 Z_1 q_0] [q_0 Z_1 q_0]$$

$$[q_0 Z_1 q_0] \rightarrow e$$

$$[q_0 Z_0 q_0] \rightarrow \varepsilon$$

Practice Question

Let $A = [q_0\ Z_0\ q_0]$ and $B = [q_0\ Z_1\ q_0]$

Then, the grammar productions will be:

$$S \rightarrow A$$

$$A \rightarrow bBA \mid \varepsilon$$

$$B \rightarrow bBB \mid e$$