Bab 6 Pushdown Automata

Definition

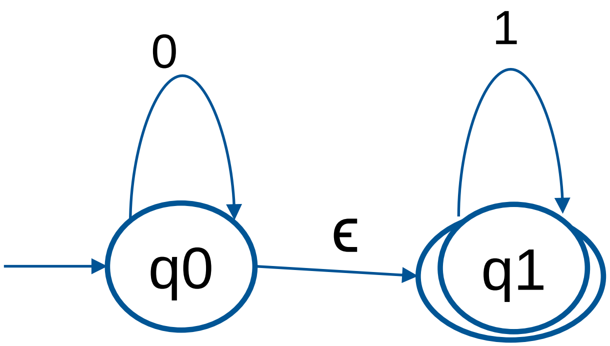
Moves of the PDA

Languages of the PDA

Deterministic PDA's

Informatika

0^*1^*



0^n1^n

000111

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.

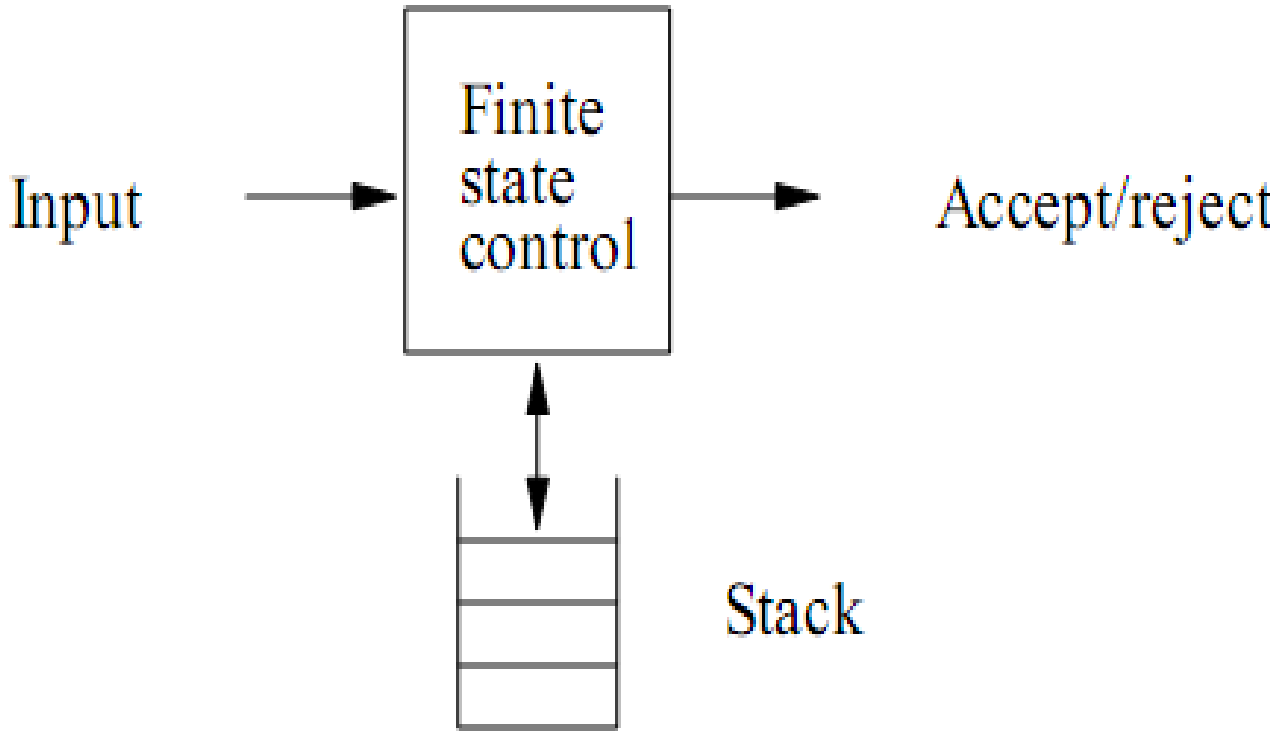
Intuition: PDA

- **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.”

Ilustrasi PDA



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.

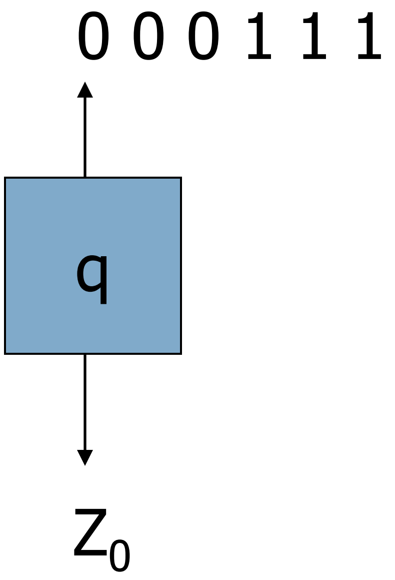
Example: PDA – (2)

- **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.

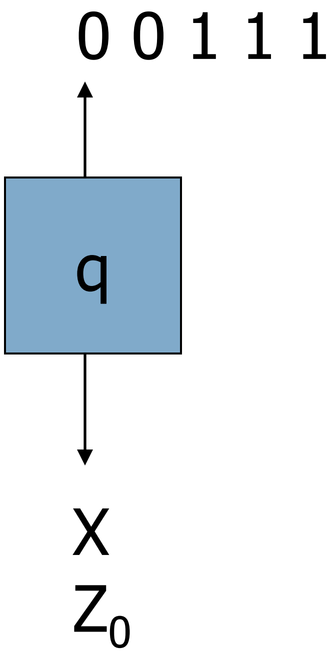
Example: PDA – (3)

- **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.

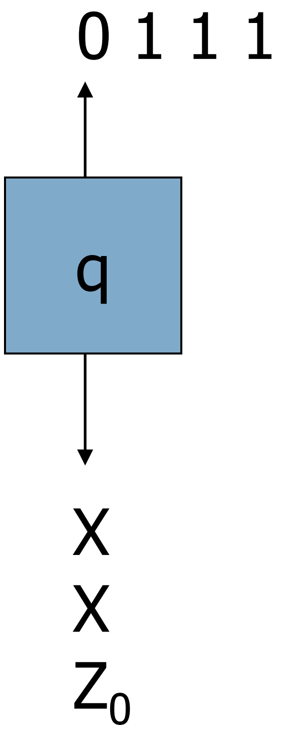
Actions of the Example PDA



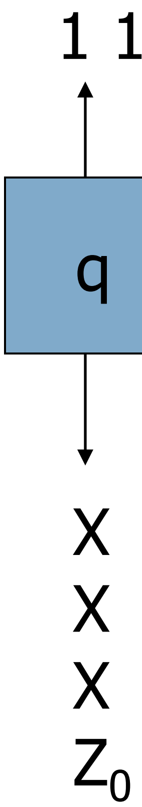
Actions of the Example PDA



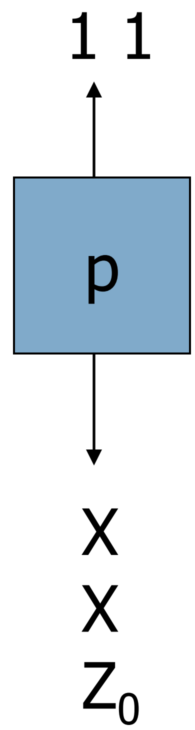
Actions of the Example PDA



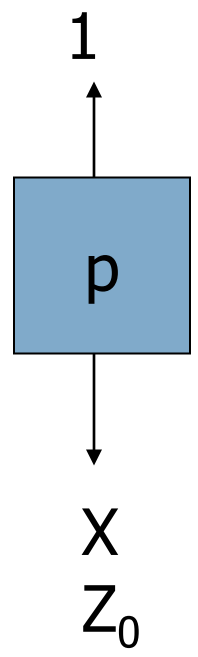
Actions of the Example PDA



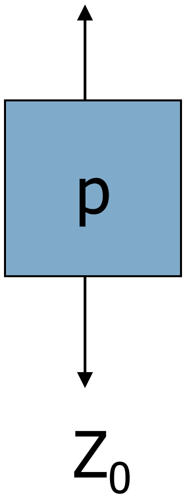
Actions of the Example PDA



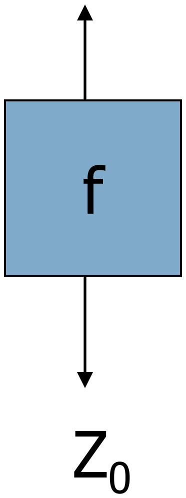
Actions of the Example PDA



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⊦J.**
- **Formally, (q, aw, Xα)⊦(p, w, βα) for any w and α, if δ(q, a, X) contains (p, β).**
- **Extend ⊦ to ⊦*, meaning “zero or more moves,” by:**
  - Basis: I⊦*I.
  - Induction: If I⊦*J and J⊦K, then I⊦*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?

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

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.**

0^n1^{2n}

$\{w\#x \mid w^R \text{ is a substring of } x, \text{ where } w, x \in \{a, b\}^*\}$

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

- **There are 2 approach to defining language of a PDA: PDA final state, PDA empty stack**
- **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.**

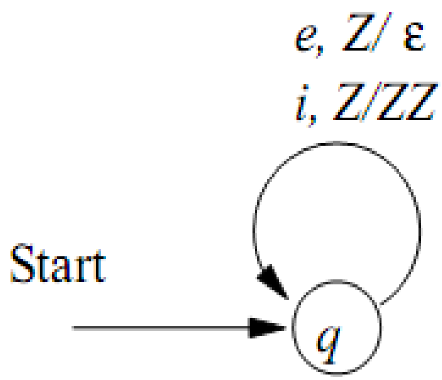
Contoh CFG if-else

Let's design P_N for for cathing errors in strings meant to be in the *if-else*-grammar G

$$S \to \epsilon | SS | iS | iSe.$$

Here e.g. $\{ieie, iie, iei\} \subseteq G$, and e.g. $\{ei, ieeii\} \cap G = \emptyset$.

The diagram for P_N is



Formally,

$$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z),$$

where $\delta_N(q, i, Z) = \{(q, ZZ)\}$,
and $\delta_N(q, e, Z) = \{(q, \epsilon)\}$.

Equivalence of Language Definitions

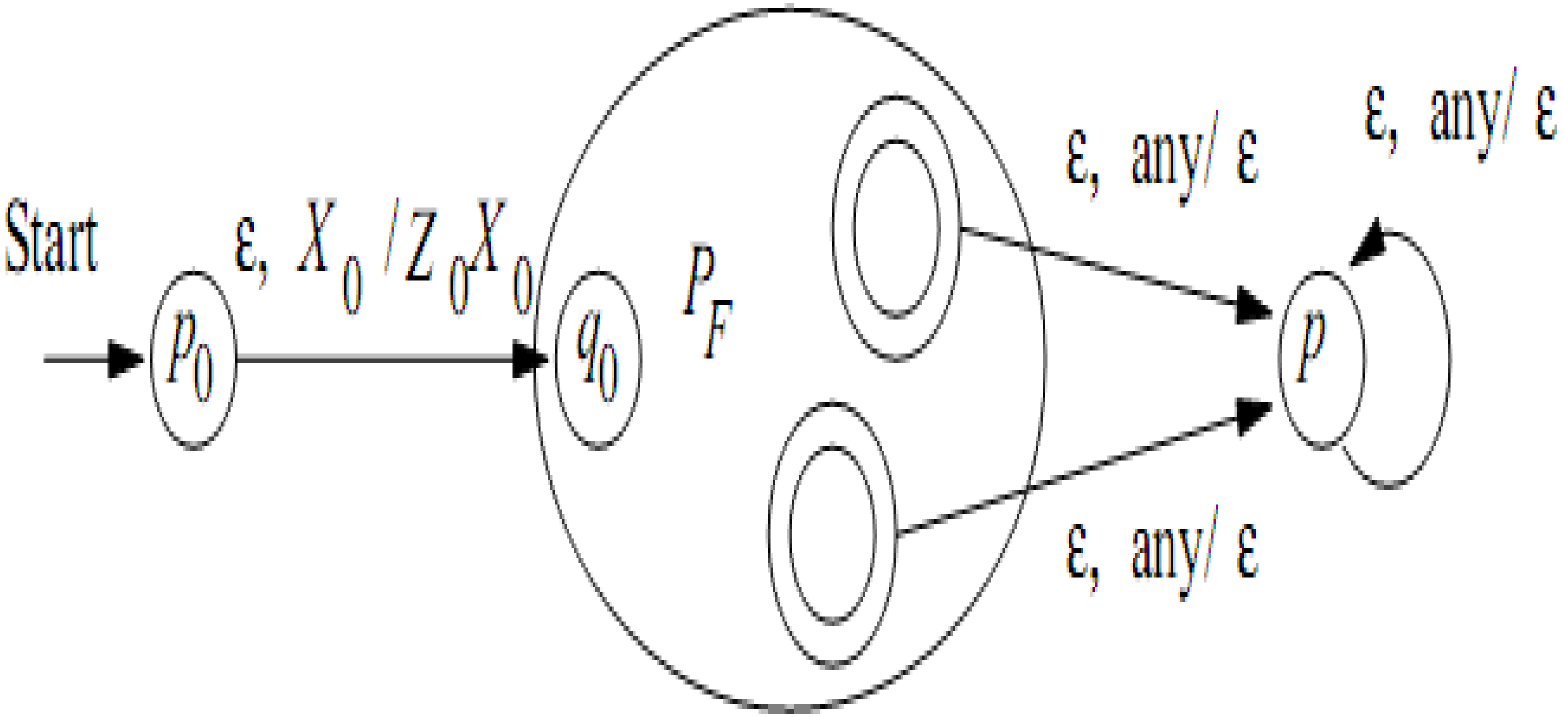
1. If L = L(P), then there is another PDA P’ such that L = N(P’).
2. If L = N(P), then there is another PDA P’’ such that L = L(P’’).

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.**

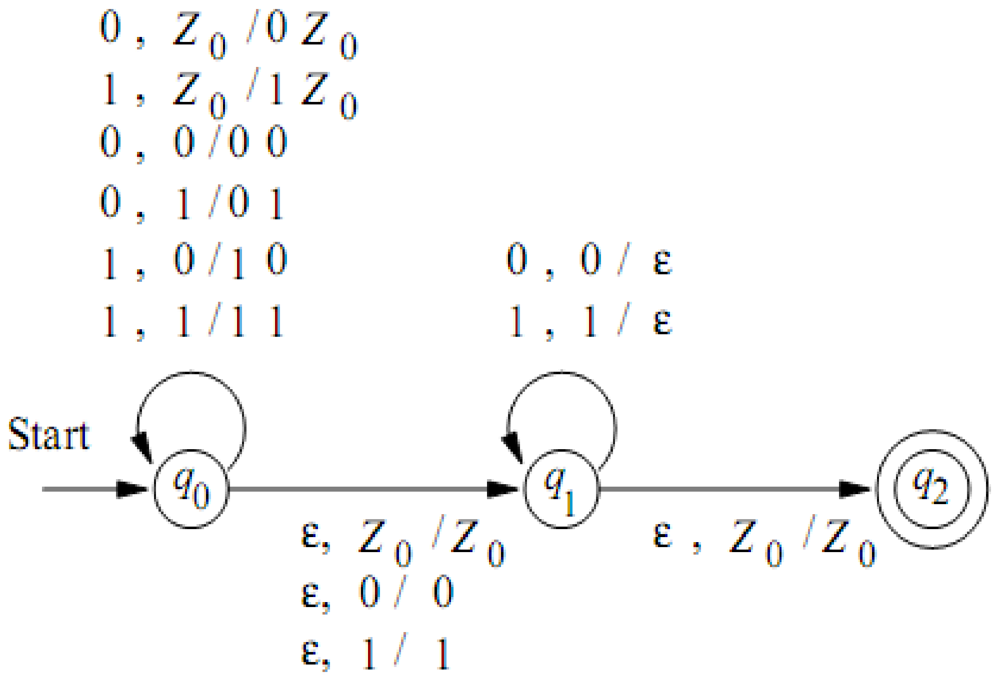
Proof: L(P) -> N(P')

- **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.


Start
ε, X0 /Z0X0
p0
q0
PF
ε, any/ ε
ε, any/ ε
ε, any/ ε
p

The PDA for L_{wwr} as a transition diagram:

Example: The PDA



$$P = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \{q_2\}),$$

where δ is given by the following table (set brackets missing):

	$0, Z_0$	$1, Z_0$	$0,0$	$0,1$	$1,0$	$1,1$	ϵ, Z_0	$\epsilon, 0$	$\epsilon, 1$
$\rightarrow q_0$	$q_0, 0Z_0$	$q_0, 1Z_0$	$q_0, 00$	$q_0, 01$	$q_0, 10$	$q_0, 11$	q_1, Z_0	$q_1, 0$	$q_1, 1$
q_1			q_1, ϵ			q_1, ϵ	q_2, Z_0		
$\star q_2$									

Example: On input 1111 the PDA

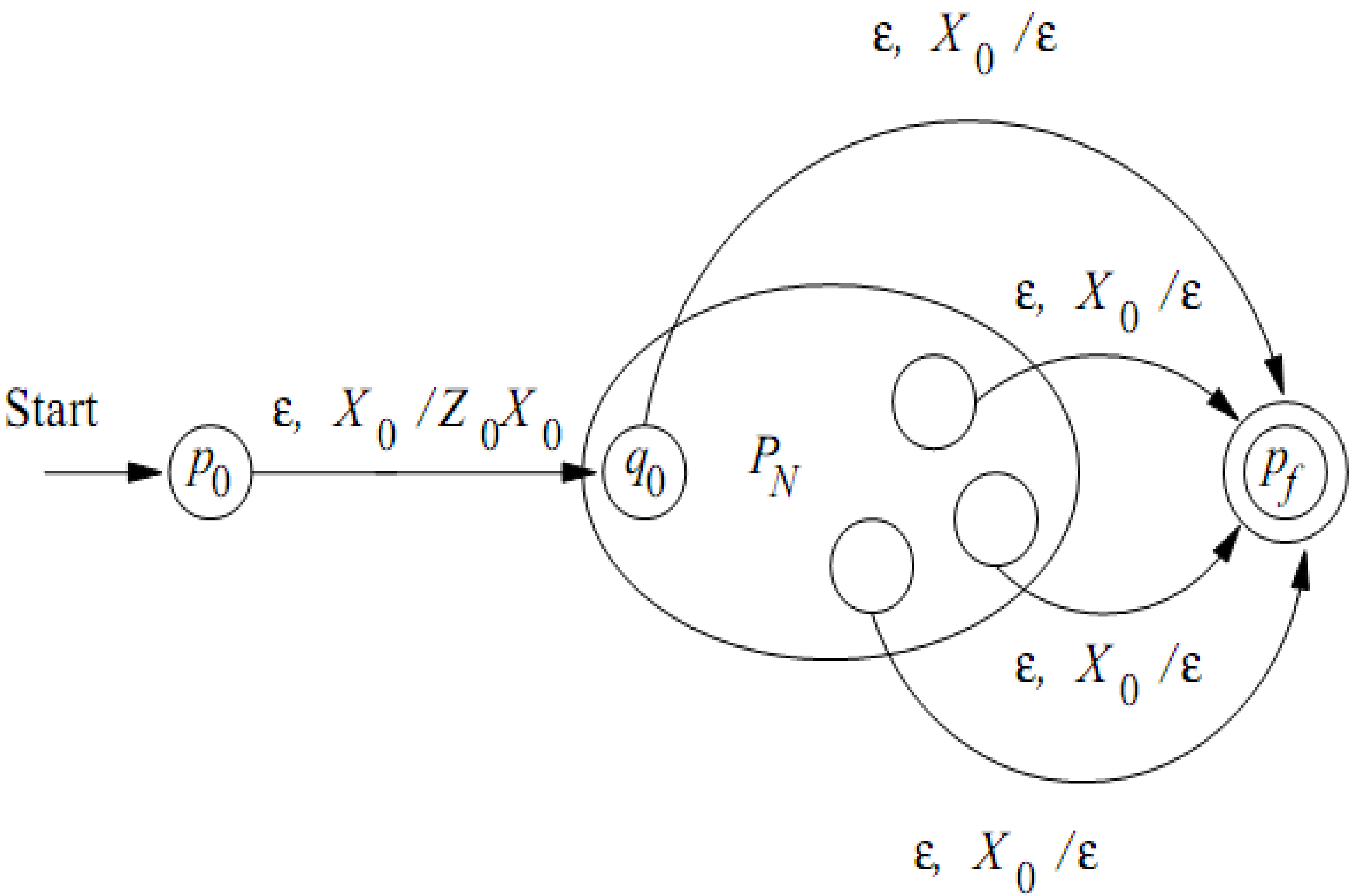
$$
\begin{tikzcd}
(q_0, 1111, Z_0) \arrow[d] \arrow[dr] & & \\
(q_0, 111, 1Z_0) \arrow[d] \arrow[dr] & (q_1, 1111, Z_0) \arrow[r] & (q_2, 1111, Z_0) \\
(q_0, 11, 11Z_0) \arrow[d] \arrow[dr] & (q_1, 111, 1Z_0) \arrow[r] & (q_1, 11, Z_0) \arrow[d] \\
(q_0, 1, 111Z_0) \arrow[d] \arrow[dr] & (q_1, 11, 11Z_0) \arrow[dr] & (q_2, 11, Z_0) \\
(q_0, \varepsilon, 1111Z_0) \arrow[d] & (q_1, 1, 111Z_0) \arrow[d] & (q_1, 1, 1Z_0) \arrow[d] \\
(q_1, \varepsilon, 1111Z_0) & (q_1, \varepsilon, 11Z_0) & (q_1, \varepsilon, Z_0) \arrow[d] \\
 & & (q_2, \varepsilon, Z_0)
\end{tikzcd}
$$

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'')

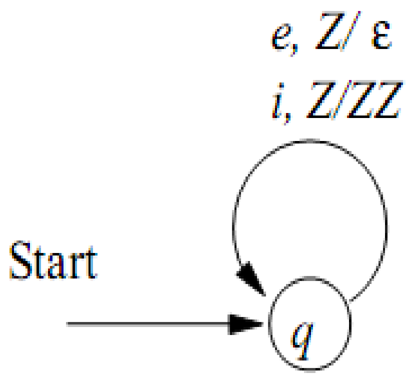
- **P'' has all the states, symbols, and moves of P, plus:**
  1. Stack symbol X_0, used to guard the stack bottom.
  2. New start state s and final state f.
  3. $\delta(s, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. Get P started.
  4. $\delta(q, \epsilon, X_0) = \{(f, \epsilon)\}$ for any state q of P.


ε, X_0 /ε
ε, X_0 /ε
Start
ε, X_0 /Z_0X_0
p_0
q_0
P_N
p_f
ε, X_0 /ε
ε, X_0 /ε

Let's design P_N for for cathing errors in strings meant to be in the *if*-*else*-grammar G

$$S \rightarrow \epsilon|SS|iS|iSe.$$

Here e.g. $\{ieie, iie, iei\} \subseteq G$, and e.g. $\{ei, ieeii\} \cap G = \emptyset$. The diagram for P_N is



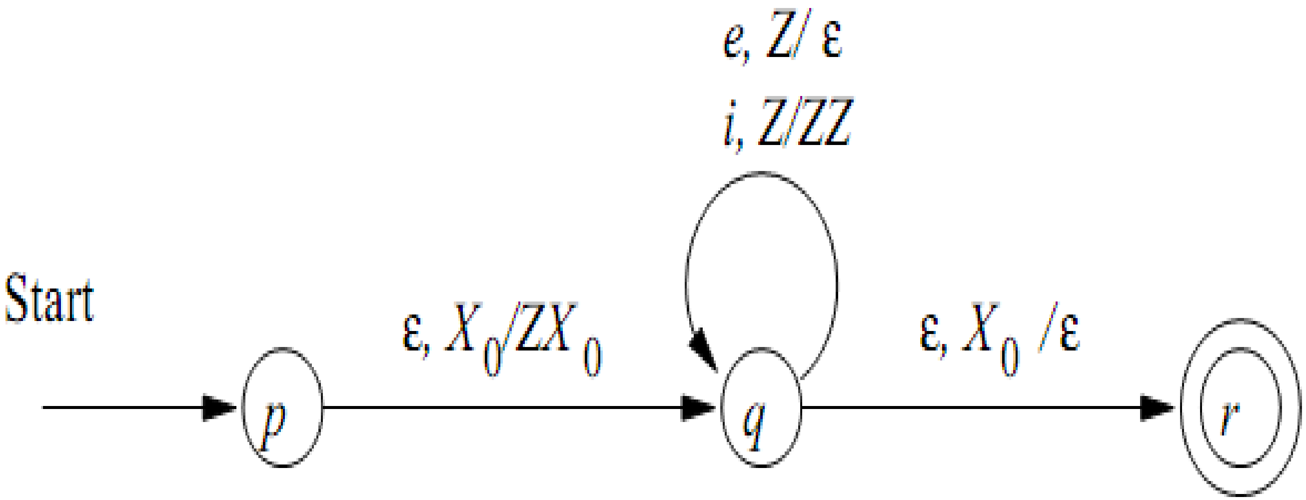
Formally,

$$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z),$$

where $\delta_N(q, i, Z) = \{(q, ZZ)\}$,

and $\delta_N(q, e, Z) = \{(q, \epsilon)\}$.

The diagram for P_F is



$$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\}),$$

where

$\delta_F(p, \epsilon, X_0) = \{(q, ZX_0)\}$,

$\delta_F(q, i, Z) = \delta_N(q, i, Z) = \{(q, ZZ)\}$,

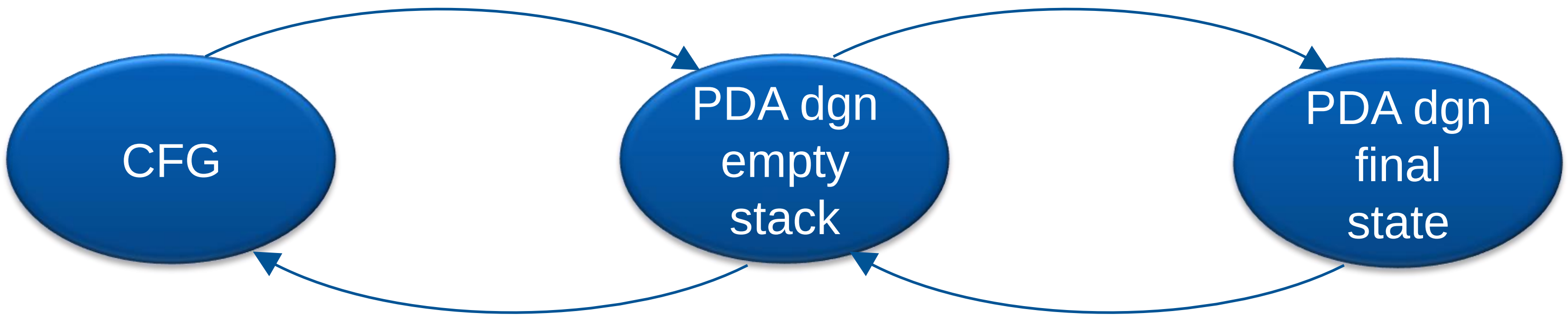
$\delta_F(q, e, Z) = \delta_N(q, e, Z) = \{(q, \epsilon)\}$, and

$\delta_F(q, \epsilon, X_0) = \{(r, \epsilon)\}$

Ekivalensi antara PDA dan CFG

Informatika

Ekivalen PDA dan CFG



CFG G = (V, T, Q, S)

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

A → 0C

Variable (non terminal)

Terminal

Stack symbol

$\boldsymbol{\delta(q_0,1,Z_0)=(q_0,1Z_0)}$

state

Input symbol

CFG ke PDA dgn Empty Stack

- **Ide Konversi, G= (V,T,P,S)**
  - semua isi pada aturan produksi di CFG G akan diproses sebagai manipulasi stack shg hanya akan ada **satu state q** pada PDA dan **banyak simbol stack**.
  - Maka arti dari **A → α** akan berarti untuk **“*top symbol of the stack* adalah A”**, **“gantilah top tsb dgn α”** atau **“pop A dan push α”**

- **maka PDA-nya (P = (Q, Σ, Γ, δ, q_0, Z_0)) adalah:**
  - P = ({q}, T, V U T, δ, q, S)
  - dimana {q} adl states(Q); T adl input symbol (Σ); V U T adl stack alphabet (Γ); δ adl transition function; q adl start state dan S adalah start stack symbol

CFG ke PDA dgn Empty Stack (2)

- **CFG G = (V, T, Q, S)**
  - V: variable (non terminal); T: terminal; Q: aturan produksi; S: start variable
- **PDA P = ({q}, T, V U T, δ, q, S)**
  - P = (Q, Σ, Γ, δ, q_0, Z_0)

    Q:state; Σ:input symbol; Γ:stack symbol; δ: transition function; q_0:start state; Z_0:start stack
- **Dengan δ adl:**
  - Utk setiap terminal a bagian dari T akan ada $\delta(q,a,a) = \{(q,\varepsilon)\}$
  - Utk setiap variable A bagian dari V, akan ada $\delta(q,\varepsilon,A) = \{(q,\beta) | A \rightarrow \beta$ adalah produksi di Q$\}$

Contoh CFG ke PDA

- **CFG = ({I, E}, {0,1,+,*},G,E) dimana G**
  - I → 0 | 1 | I0 | I1
  - E → I | E + E | E * E
- **PDA = ({q}, T, V U T, δ, q, S) = ({q}, {0,1,+,*}, {I,E,0,1,+,*}, δ, q, E) dgn δ adl**
  - Utk setiap terminal: δ(q,0,0) = (q, ε); δ(q,1,1)=(q,ε); δ(q,+,+) = (q, ε); δ(q,*,*) = (q, ε)
  - Utk setiap non terminal:
    - δ(q, ε,I) = {(q, 0), (q, 1), (q,I0), (q,I1)}
    - δ(q, ε,E) = {(q, I), (q, E+E), (q,E*E)}

Latihan

- **6.3.1**
  - Konversi grammar berikut ke PDA empty stack:

    $S \rightarrow 0\ S\ 1 \mid A$

    $A \rightarrow 1\ A\ 0 \mid S \mid \epsilon$

$L = \{a^n b^m c^k : k \neq n + m\}$

Let $L1 = \{a^n b^m c^k : k < n + m\}$, $L2 = \{a^n b^m c^k : k > n + m\}$

$S \to EcC \mid aAE \mid AU$

$A \to aA \mid \lambda$

$B \to bB \mid \lambda$

$C \to cC \mid \lambda$

$E \to aEc \mid F$

$F \to bFc \mid \lambda$

$U \to aUc \mid V$

$V \to bVc \mid bB$

Q={q}

$\partial(q,a,a) = \{(q,\epsilon)\}$

$\partial(q,b,b) = \{(q,\epsilon)\}$

$\partial(q,c,c) = \{(q,\epsilon)\}$

$\partial(q,\epsilon,S) = \{(q,EcC),(q,aAE),(AU)\}$

$\partial(q,\epsilon,A) = \{(q,aA),(q, \epsilon)\}$

$\partial(q,\epsilon,B) = \{(q,bB),(q, \epsilon)\}$

$\partial(q,\epsilon,C) = \{(q,cC),(q, \epsilon)\}$

$\partial(q,\epsilon,E) = \{(q,aEc),(q, F)\}$

$\partial(q,\epsilon,F) = \{(q,bFc),(q, \epsilon)\}$

$\partial(q,\epsilon,U) = \{(q,aUc),(q, V)\}$

$\partial(q,\epsilon,V) = \{(q,bVc),(q, bB)\}$

$$L = \{a^n b^m c^k : k \neq n + m\}$$

$$L = \{a^n b^m c^k : k \neq n + m\}$$

PDA (P) ke CFG (G)

- **Misalkan P = (Q, Σ, Γ, δ, q_0, Z_0)**
- **Maka G = (V, Σ, R, S) dimana:**
  - simbol Terminal adalah sama dengan input symbol pada PDA P yaitu Σ;
  - Ditambahkan satu non terminal **S** sbg start symbol
  - Simbol non terminal V adalah S ditambah dengan hasil kombinasi antara simbol stack dan state sbb: [pXq] dimana p dan q adalah state dan X adalah stack. Pada akhirnya, [pXq] ini bisa diganti dgn satu karakter spt misalnya A.

Ide Dasar PDA ke CFG

- **Non Terminal (CFG)**
  - Berdasar pada transisi pada PDA
    - Transisi dari satu state asal ke state tujuan
      - state asal bisa sama dengan state tujuan
    - Aksi pada stack
      - Pop
      - Push
      - Tetap
  - [q Z p] yg artinya dari mewakili transisi dari state q ke state p dgn aksi pada top of stack berupa Z
    - Tulis semua kombinasi state dan stack yang mungkin untuk semua non terminal pada CFG
    - Contoh: [qZp], [qZq], [pZp], [pZq]
  - Target adalah [q_0 Z_0 q_f] yaitu transisi dari state awal q_0 ke state final q_f dengan mem-pop Z_0 start stack
    - karena PDA dgn empty stack, maka q_f diganti dgn semua state yg ada

Aturan Produksi (1)

- **Utk semua state p pada P, buat aturan produksi sbb S → [q_0 Z_0 p], dimana q_0 adalah start state pada P dan Z_0 adalah start stack pada P. Jumlah aturan produksi ini sama dengan jumlah state yang ada pada P. Misalkan selain p, ada state q, maka akan ada aturan S → [q_0 Z_0 q]**

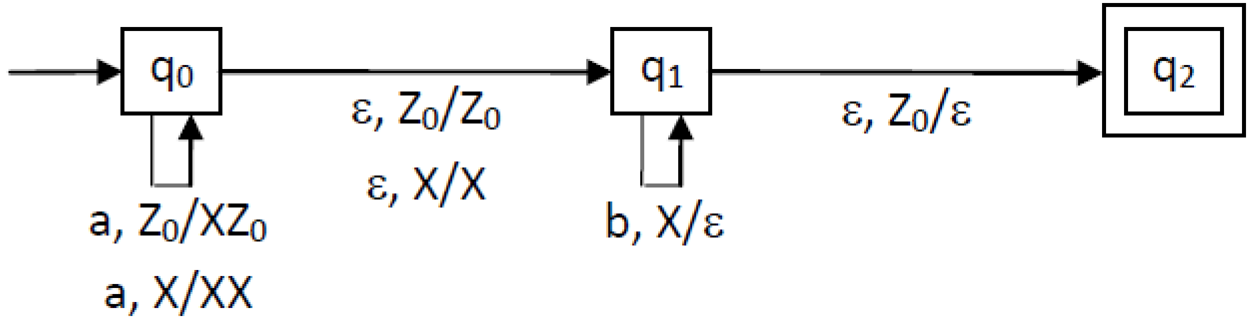
Aturan Produksi (2)

- **Target adalah S → [q_0 Z_0 q_f]**
- **Prinsip CFG: mengurai dari solusi total ke sub-nya → pohon ke sub pohon dst**
- **Untuk setiap fungsi transisi pada PDA, buatlah aturan produksi sbb:**
  - Jika transisi di pop:
    - δ(q,0,Z) = (p,ϵ) maka pd CFG: [qZp] → 0
  - Jika transisi tetap atau diganti:
    - δ(q,0,Z) = (p,Y) maka pd CFG: [qZ*r*] → 0 [pY*r*] dimana r adalah semua state yg ada pd PDA
  - Jika transisi di push:
    - δ(q,0,Z) = (p,YZ) maka [qZ*r*] → 0 [pY***s***] [***s***Z*r*] dimana r dan s adalah semua state yg ada pd PDA

Aturan Produksi (3)

- **Untuk fungsi transisi $\delta(q,a,X)=(r, Y_1Y_2...Y_k)$, maka buatlah aturan produksi untuk setiap kombinasi state sbb:**
  - $[q\ X\ \Delta] \rightarrow a\ [r\ Y_1\ ♣][♣\ Y_2\ ...]\ ...\ [...\ Y_k\ \Delta]$

Contoh: untuk $\{a^n b^n\}$



- **$S \rightarrow [q_0\ Z_0\ q_2]$**
- **$[q_1\ Z_0\ q_2] \rightarrow \epsilon$**
- **$[q_1\ X\ q_1] \rightarrow b$**

$[q_0\ Z_0\ q_0] \rightarrow [q_1\ Z_0\ q_0]$ $\qquad$ $[q_0\ X\ q_0] \rightarrow [q_1\ X\ q_0]$

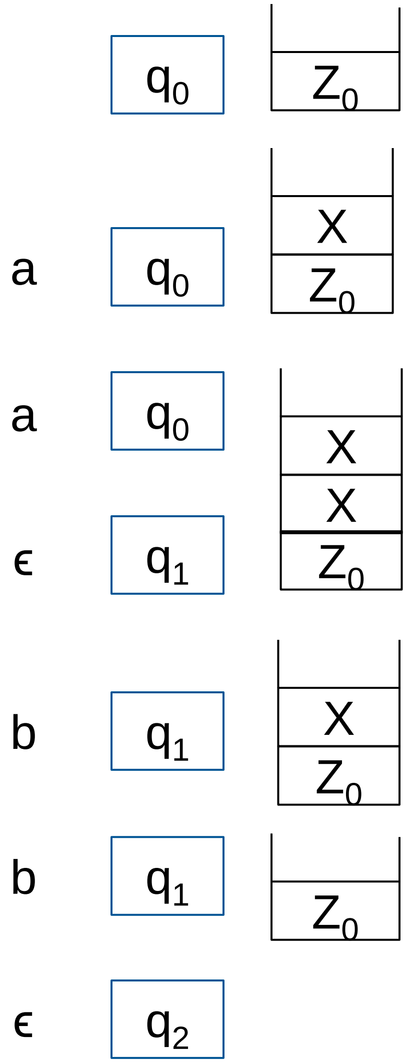
$[q_0\ Z_0\ q_1] \rightarrow [q_1\ Z_0\ q_1]$ $\qquad$ $[q_0\ X\ q_1] \rightarrow [q_1\ X\ q_1]$

$[q_0\ Z_0\ q_2] \rightarrow [q_1\ Z_0\ q_2]$ $\qquad$ $[q_0\ X\ q_2] \rightarrow [q_1\ X\ q_2]$

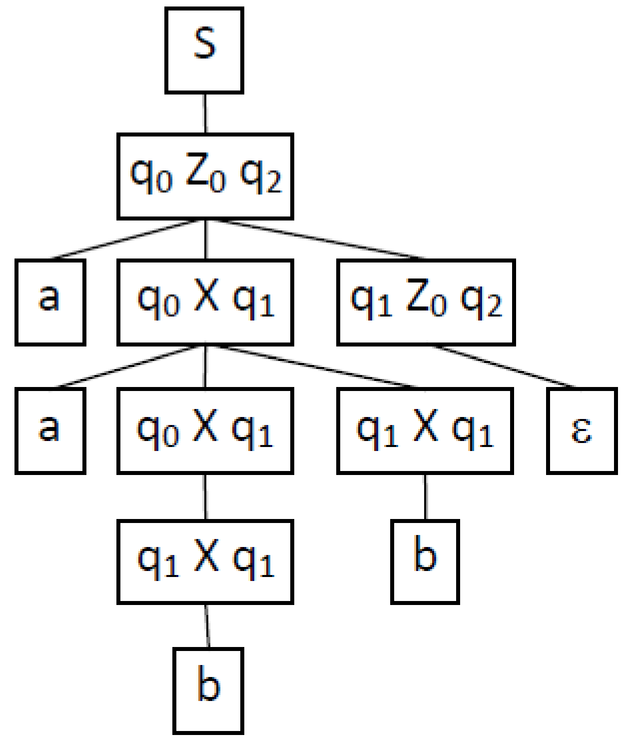
$$[q_0\ Z_0\ q_0] \rightarrow a\ [q_0\ X\ q_0]\ [q_0\ Z_0\ q_0] \mid a\ [q_0\ X\ q_1]\ [q_1\ Z_0\ q_0] \mid a\ [q_0\ X\ q_2]\ [q_2\ Z_0\ q_0]$$
$$[q_0\ Z_0\ q_1] \rightarrow a\ [q_0\ X\ q_0]\ [q_0\ Z_0\ q_1] \mid a\ [q_0\ X\ q_1]\ [q_1\ Z_0\ q_1] \mid a\ [q_0\ X\ q_2]\ [q_2\ Z_0\ q_1]$$
$$[q_0\ Z_0\ q_2] \rightarrow a\ [q_0\ X\ q_0]\ [q_0\ Z_0\ q_2] \mid a\ [q_0\ X\ q_1]\ [q_1\ Z_0\ q_2] \mid a\ [q_0\ X\ q_2]\ [q_2\ Z_0\ q_2]$$

$$[q_0\ X\ q_0] \rightarrow a\ [q_0\ X\ q_0]\ [q_0\ X\ q_0] \mid a\ [q_0\ X\ q_1]\ [q_1\ X\ q_0] \mid a\ [q_0\ X\ q_2]\ [q_2\ X\ q_0]$$
$$[q_0\ X\ q_1] \rightarrow a\ [q_0\ X\ q_0]\ [q_0\ X\ q_1] \mid a\ [q_0\ X\ q_1]\ [q_1\ X\ q_1] \mid a\ [q_0\ X\ q_2]\ [q_2\ X\ q_1]$$
$$[q_0\ X\ q_2] \rightarrow a\ [q_0\ X\ q_0]\ [q_0\ X\ q_2] \mid a\ [q_0\ X\ q_1]\ [q_1\ X\ q_2] \mid a\ [q_0\ X\ q_2]\ [q_2\ X\ q_2]$$

- **aabb**



q0
Z0
a
q0
X
Z0
a
q0
X
X
ε
q1
Z0
b
q1
X
Z0
b
q1
Z0
ε
q2



S
q0 Z0 q2
a
q0 X q1
q1 Z0 q2
a
q0 X q1
q1 X q1
ε
q1 X q1
b
b

Latihan

* **Exercise 6.3.3:** Convert the PDA $P = (\{p, q\}, \{0, 1\}, \{X, Z_0\}, \delta, q, Z_0)$ to a CFG, if δ is given by:

1. $\delta(q, 1, Z_0) = \{(q, XZ_0)\}$.
2. $\delta(q, 1, X) = \{(q, XX)\}$.
3. $\delta(q, 0, X) = \{(p, X)\}$.
4. $\delta(q, \epsilon, X) = \{(q, \epsilon)\}$.
5. $\delta(p, 1, X) = \{(p, \epsilon)\}$.
6. $\delta(p, 0, Z_0) = \{(q, Z_0)\}$.

Contoh Soal 6.3.3

- **Dengan 2 state yaitu p dan q maka dari start symbol S , start state q dan start stack symbol Zo, diperoleh aturan produksi:**
  - S → [q Zo q]
  - S → [q Zo p]

Contoh Soal 6.3.3

- **Dari transition function 1 yaitu δ(q,1,Zo)=(q,X Zo) , diperoleh aturan produksi:**
  - [q Zo q] → 1 [q X q] [q Zo q]
  - [q Zo q] → 1 [q X p] [p Zo q]
  - [q Zo p] → 1 [q X q] [q Zo p]
  - [q Zo p] → 1 [q X p] [p Zo p]

Contoh Soal 6.3.3

- **Dari transition function 3 yaitu $\delta(q,0,X)=(p,X)$, diperoleh aturan produksi:**
  - [q X q] → 0 [p X q]
  - [q X p] → 0 [p X p]
- **Dari transition function 4 yaitu $\delta(q,\varepsilon,X)=(q,\varepsilon)$, diperoleh aturan produksi:**
  - [q X q] → ε

6.3.3

- In the following, *S* is the start symbol, *e* stands for the empty string, and *Z* is used in place of Z_0.
  - *S -> [qZq] | [qZp]*
- The following four productions come from rule (1).
  - *[qZq] -> 1[qXq][qZq]*
  - *[qZq] -> 1[qXp][pZq]*
  - *[qZp] -> 1[qXq][qZp]*
  - *[qZp] -> 1[qXp][pZp]*
- The following four productions come from rule (2).
  - *[qXq] -> 1[qXq][qXq]*
  - *[qXq] -> 1[qXp][pXq]*
  - *[qXp] -> 1[qXq][qXp]*
  - *[qXp] -> 1[qXp][pXp]*
- The following two productions come from rule (3).
  - *[qXq] -> 0[pXq]*
  - *[qXp] -> 0[pXp]*
- The following production comes from rule (4).
  - *[qXq] -> e*
- The following production comes from rule (5).
  - *[pXp] -> 1*
- The following two productions come from rule (6).
  - *[pZq] -> 0[qZq]*

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.

$$L_{wcwr} = \{wcw^R : w \in \{0,1\}^*\}$$

Then L_{wcwr} is recognized by the following DPDA