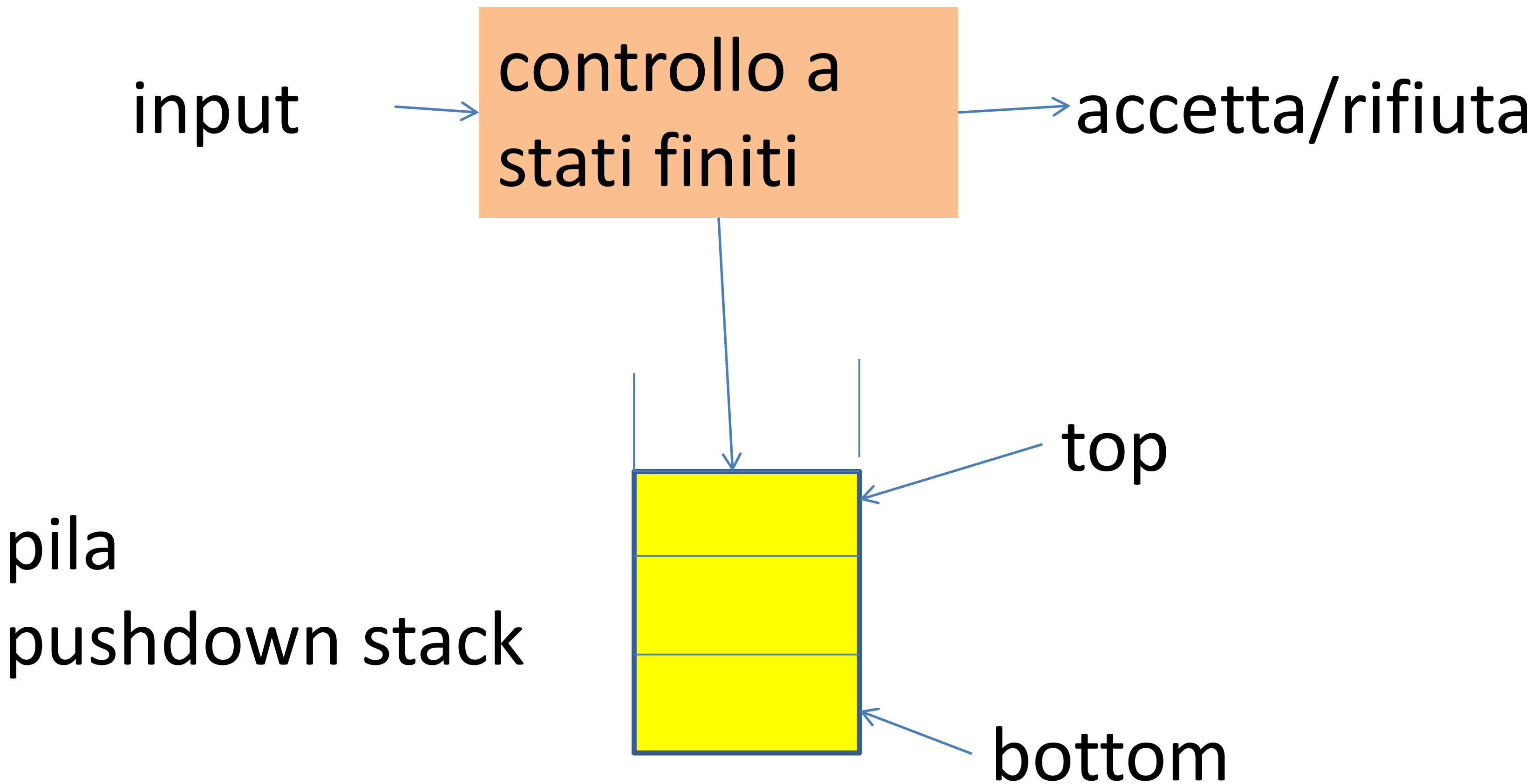
Automi a pila
PushDown Automaton (PDA)

PDA


input
controllo a
stati finiti
accetta/rifiuta
top
pila
pushdown stack
bottom

la pila può crescere arbitrariamente, ma può essere usata solo in questo modo:

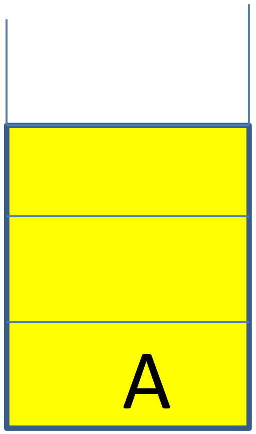
-- si guarda la cima delle pila e poi....

→la si lascia com'è

→push: si inseriscono nuove cose in cima alla pila

→pop: si toglie la cima della pila

restrizione



per vedere A devo togliere gli elementi che sono sopra A e quindi li perdo il controllo a stati finiti :

-legge il prossimo input
-guarda il simbolo in cima alla pila
-fa una transizione in cui può:
----cambiare stato (o no)
----consumare l'input (o no con ε)
----eliminare, tenere o cambiare la cima della pila

esempio:

Lww^r={ ww^r | w in {0,1}*}, sono i palindromi pari

PDA che accetta Lww^r:

--uno stato di partenza q0 che scorre l'input e lo copia sullo stack e ad ogni passo può,

--sia continuare

--sia invece indovinare di aver percorso metà dell'input (nella pila c'è w^r) e quindi iniziare il match del resto dell'input (w^r) contro lo stack

➔ nondeterministico

se alla fine dell'input, lo stack è vuoto allora OK

Definizione di PDA:

$P=(Q, \Sigma, \Gamma, \delta, q0, Z_0, F)$

--Q = insieme finito di stati, con q0 stato iniziale
-- Σ insieme finito di simboli di input
-- Γ insieme finito di alfabeto dello stack, con Z_0 simbolo iniziale,
--F contenuto in Q sono gli stati finali

-- δ è la funzione di transizione che riceve come argomento una tripla (q, a, X) dove

-q è uno stato,

-a è l'input corrente (o ε),

-X è il simbolo in cima della pila (sempre non vuota per applicare δ)

$\delta(q,a/\varepsilon,X)$ è un insieme finito di coppie (p,γ), dove p è uno stato e γ una stringa in Γ^* che rimpiazza X. Se γ è vuota allora si fa un pop, se $\gamma=X$ allora lo stack non cambia e altrimenti si fa un push.

PDA per $Lww^r = \{ ww^r \mid w \text{ in } \{0,1\}^* \}$

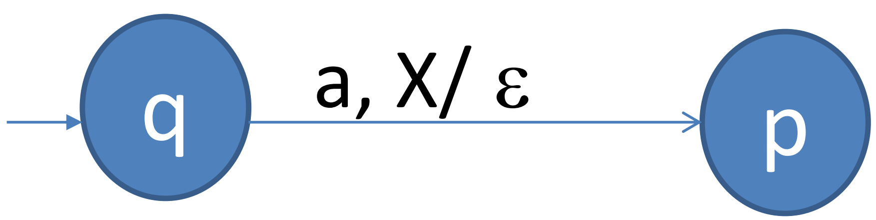
$P=(\{q0,q1,q2\},\{0,1\},\{0,1,Z0\}, \delta, q0, Z0, \{q2\})$
con δ come segue:
-- $\delta(q0,0,Z0)=\{(q0,0Z0)\}$ e $\delta(q0,1,Z0)=\{(q0,1Z0)\}$

-- $\delta(q0,0,0)=\{(q0,00)\}$, $\delta(q0,1,0)=\{(q0,10)\}$ ecc.
--$\delta(q0,\varepsilon,Z0)=\{(q1,Z0)\}$,
$\delta(q0,\varepsilon,0)=\{(q1,0)\}$ e $\delta(q0,\varepsilon,1)=\{(q1,1)\}$

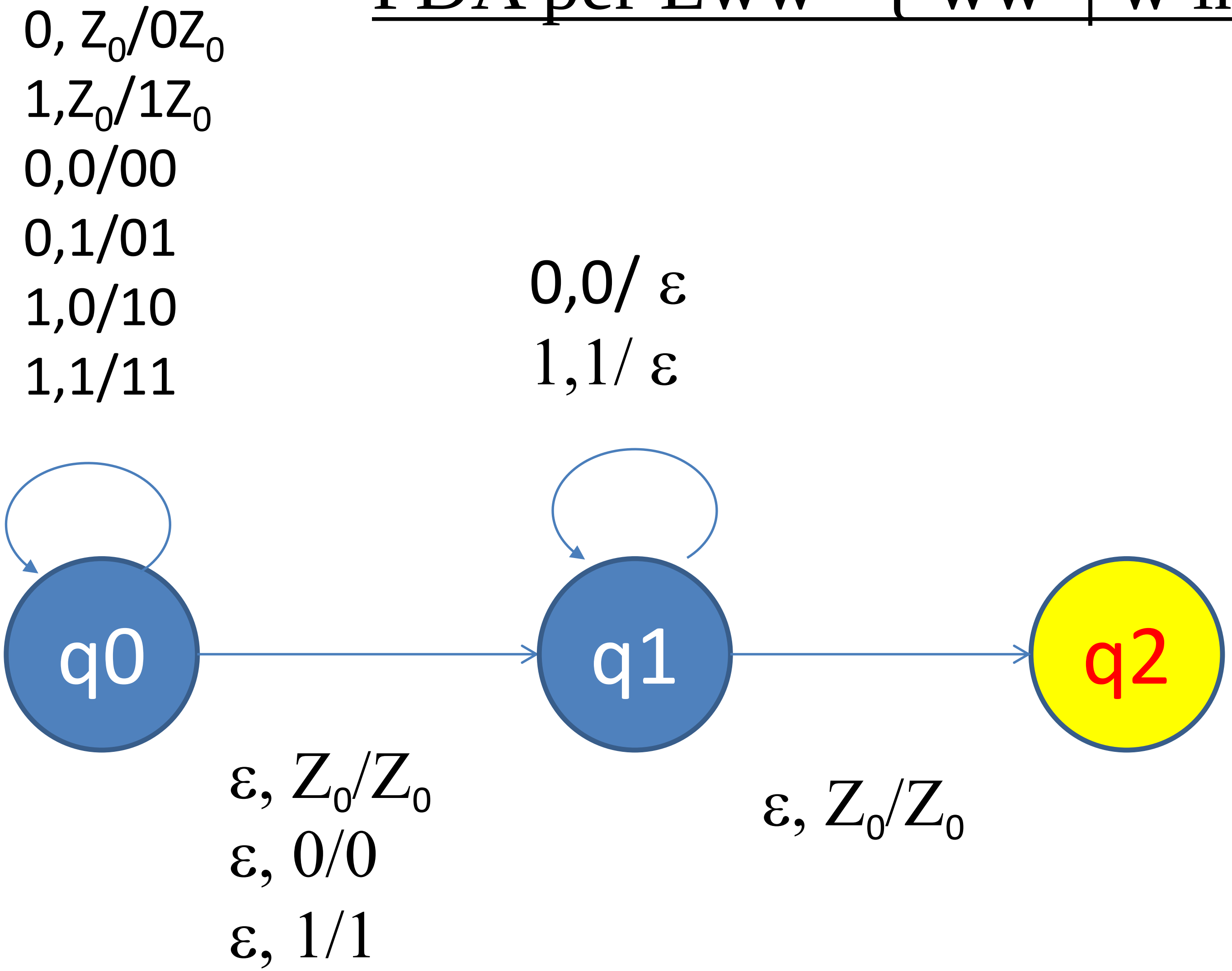
-- $\delta(q1,0,0)=\{(q1, \varepsilon)\}$, $\delta(q1,1,1)=\{(q1, \varepsilon)\}$
-- $\delta(q1, \varepsilon, Z0)=\{(q2, Z0)\}$,

notazione grafica per PDA

-nodi corrispondono agli stati
-si distingue lo stato iniziale con una freccia
-gli archi corrispondono alle transizioni e hanno etichette che rappresentano cosa succede su input e stack:
se $\delta(q,a,X)$ contiene (p, ε) allora:



PDA per $L_{ww^r} = \{ ww^r \mid w \text{ in } \{0,1\}^* \}$



Descrizioni istantanee (ID)

supponiamo che $\delta(q,a,X)$ contenga (p,α), allora

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

come al solito rappresentiamo la chiusura di $\vdash$ come $\vdash^*$

calcolo del PDA di ww^r

intuizione: posso aggiungere stringhe non usate all'input e allo stack, mantenendo la computazione (simile a context freeness)

Teorema 6.5

dato un PDA P, se $(q,x,\alpha) \vdash^* (p,y,\beta)$, allora per ogni stringa w e γ è vero che

$$(q,xw,\alpha\gamma) \vdash^* (p,yw,\beta\gamma)$$

l'inverso è falso

però vale per l'input:

Teorema 6.6

se (q,xw,α) |-* (p, yw,β)
allora è vero che
(q,x,α) |-* (p, y,β)

Dimostrazione: non si può rigenerare l'input, quindi w veramente non influenza il calcolo

Modalità di accettazione:

--per stato finale:

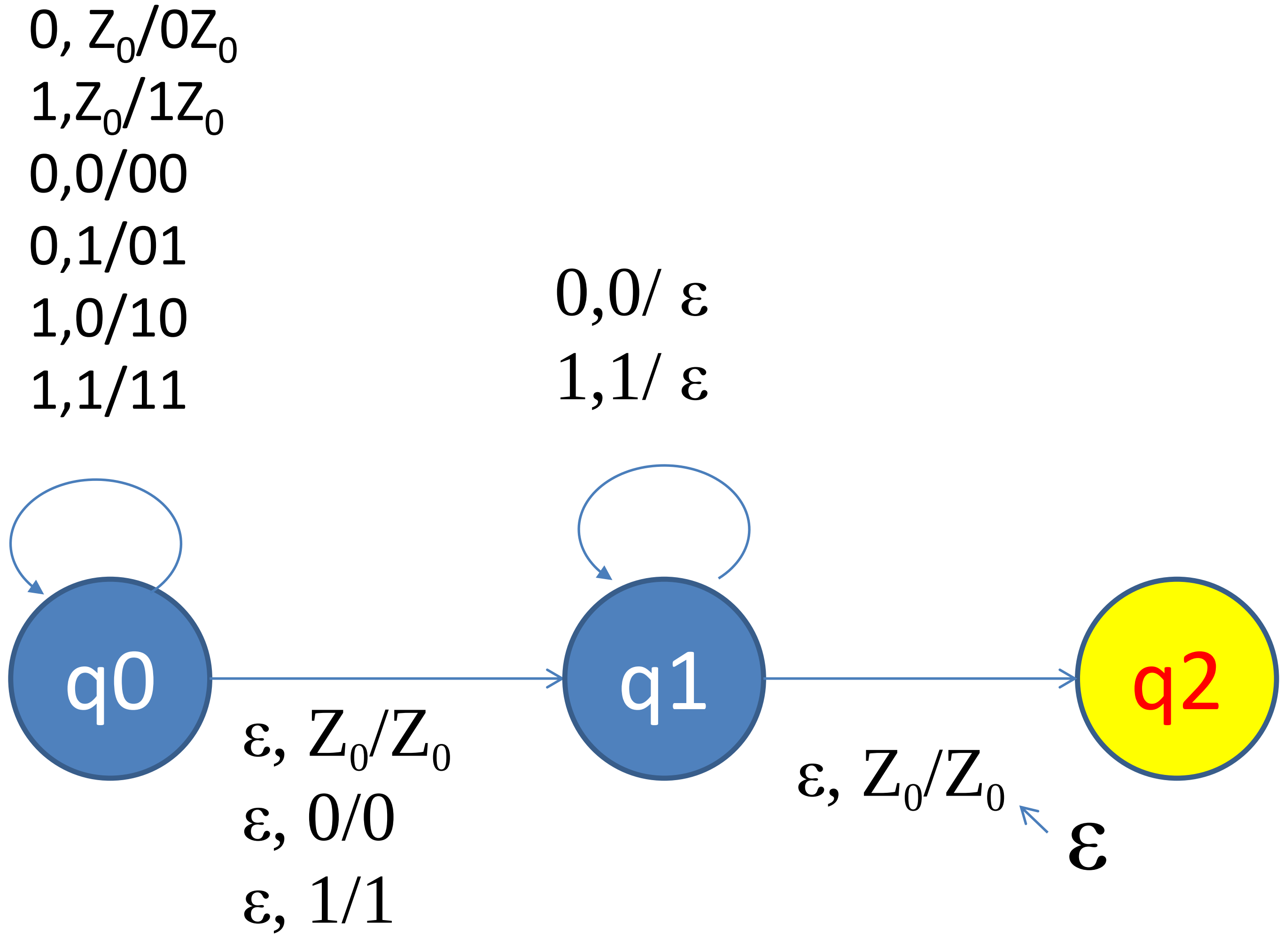
Dato P, L(P) è { w | w in Σ*, (q0,w,Z0) |- (qf,ε,α), con qf stato finale}

--con stack vuoto

N(P)={w | w in Σ*, (q0,w,Z0) |-* (q,ε,ε)}

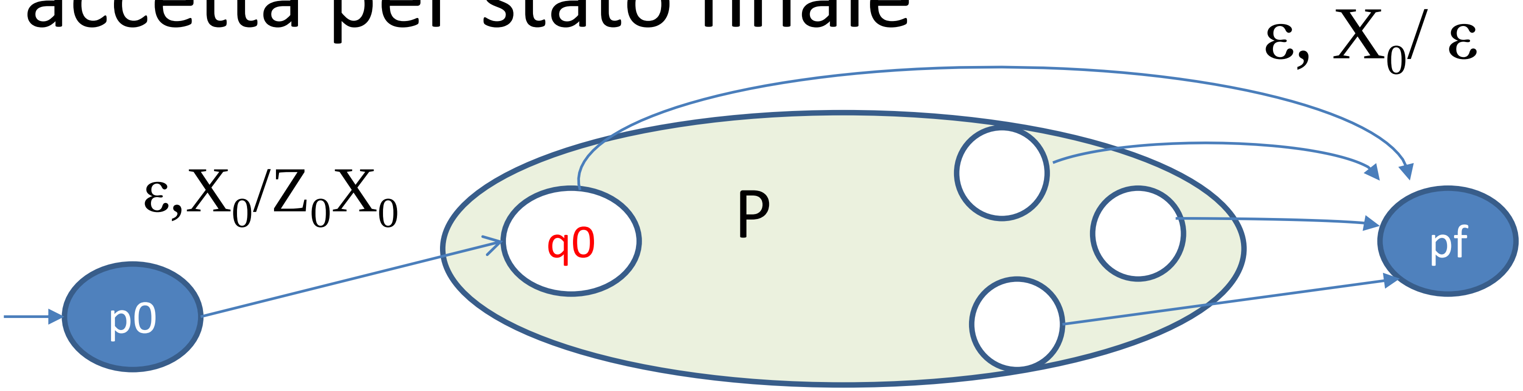
per un dato PDA P, L(P) e N(P) possono essere diversi

Il PDA che accetta ww^R, i palindromi di lunghezza pari, ha N(P)= Ø, ma è facile modificarlo



è sempre possibile passare da un PDA P che accetta in uno dei due modi ad un'altro P' che accetta nell'altro modo e accetta lo stesso linguaggio di P.

da P che accetta per stack vuoto a P' che accetta per stato finale



Dimostrazione:

w è in N(P) sse è in L(P')

(=>)

esiste $(q_0,w,Z_0) \vdash^* (q, \varepsilon, \varepsilon)$ per il Teorema 6.5,

$(q_0,w,Z_0X_0) \vdash^* (q, \varepsilon,X_0)$

e per costruzione esiste

$(q, \varepsilon,X_0) \vdash (qf, \varepsilon, \varepsilon)$

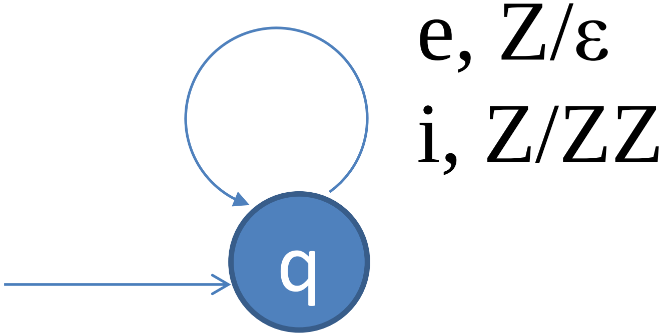
infine esiste: $(p0,w,X_0) \vdash (q0,w, Z_0X_0)$

(<=) per costruzione una computazione di P' è:

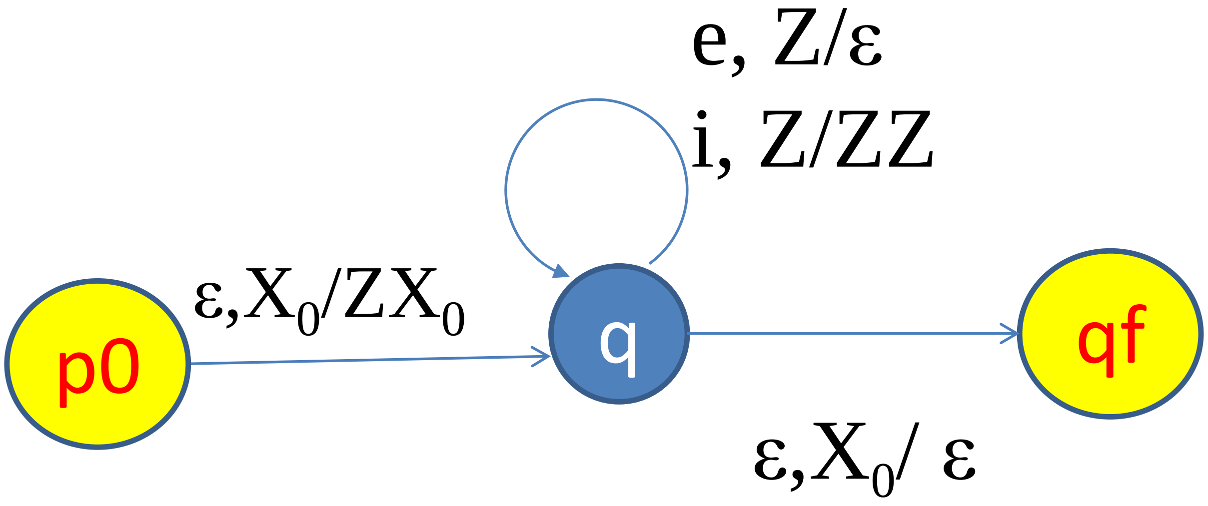
$(p0,w,X_0) \vdash (q0,w,Z_0X_0) \vdash^* (q,\varepsilon,X_0) \vdash (pf, \varepsilon, \varepsilon)$

dove questa è una computazione di P

<u>esempio</u>: un PDA che accetta con stack vuoto le stringhe in {i,e}* tali che il numero di e supera quello degli i (insomma sono i programmi sbagliati)



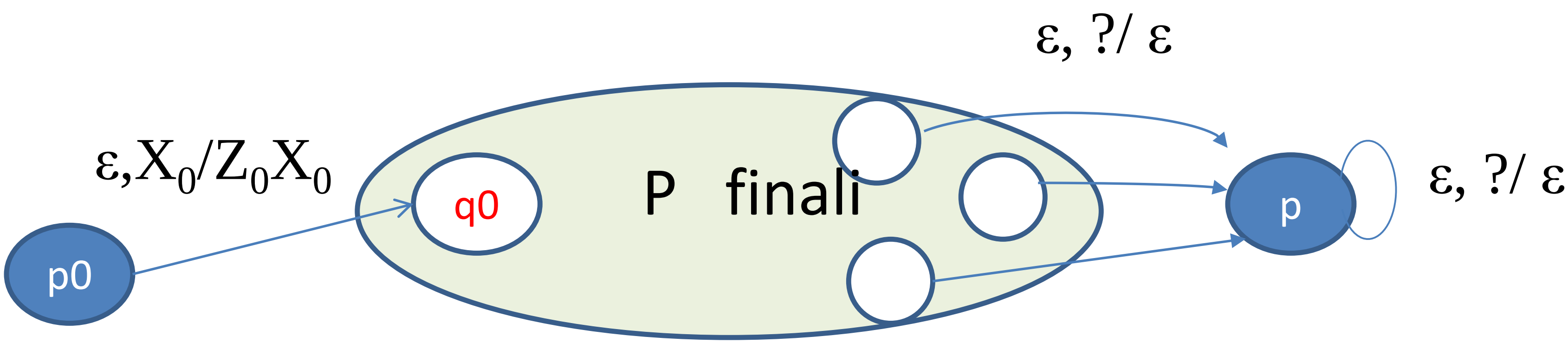
attenzione: Z_0 è Z


e, Z/ε
i, Z/ZZ
ε,X0/ZX0
p0
q
qf
ε,X0/ ε

ma questi 2 PDA accettano proprio quello che vorremmo ?

differenza tra i 2 modi di accettazione

Da stato finale P a stack vuoto P’



X_0 previene che P svuoti lo stack inavvertitamente (che diventerebbe accettazione in P’), visto che P non ha mosse per X_0

(N(P’) include L(P)) se $(q0,w,Z_0)|\text{-}^*(qf, \varepsilon,\alpha)$ allora
$(p0,w,X_0)|\text{-}(q0,w,Z_0X_0)|\text{-}^*(qf, \varepsilon,\alpha X_0)|\text{-}^*(p, \varepsilon, \varepsilon)$
(L(P) include N(P)) è l’inversa