Modelli operazionali non deterministici (1)

- Di solito si pensa a un algoritmo come a una sequenza di operazioni ben determinata
  - Dato uno stato e un ingresso, non c'è dubbio sul prossimo passo
- Esempio: confrontiamo

```
if x>y then max:=x else max:=y
```

con

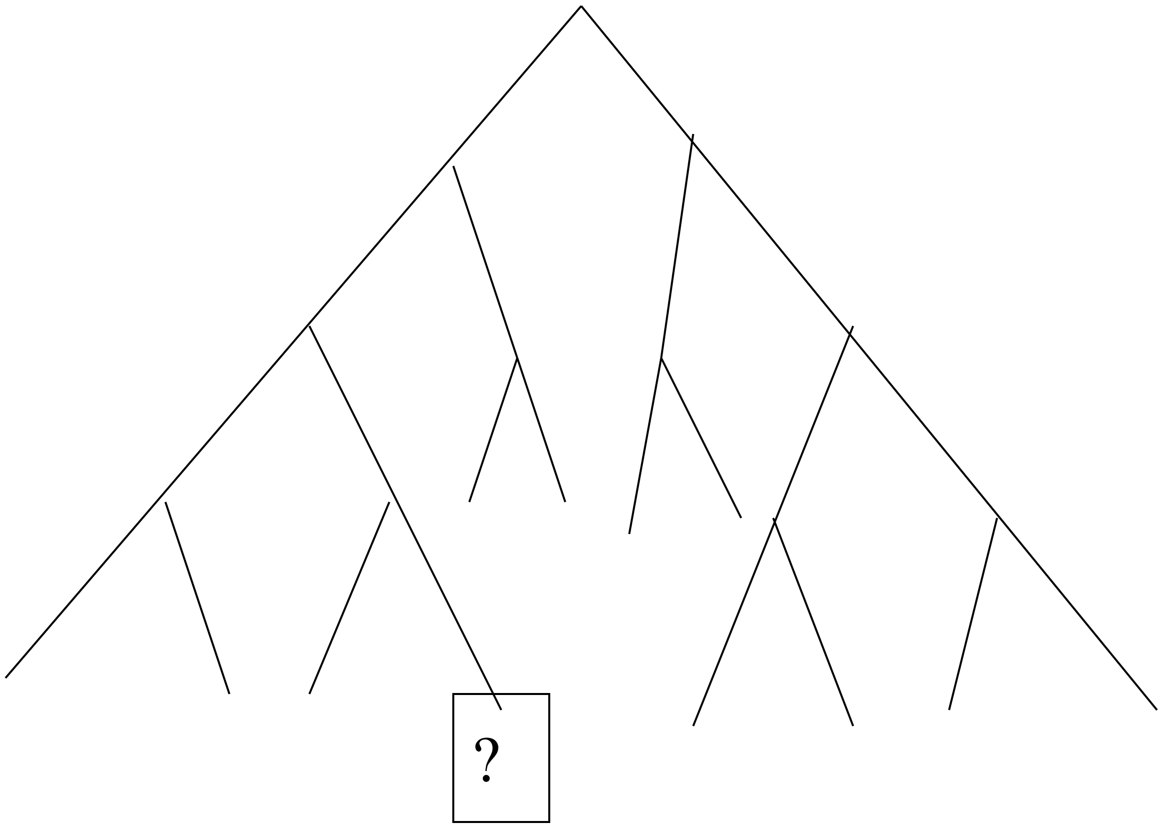
```
if
        x>=y then max:=x
        x<=y then max:=y
fi
```

Modelli operazionali non deterministici (2)

- E' solo una questione di eleganza?
- Prendiamo il costrutto **`case`** del Pascal: perché non considerare anche qualcosa del genere?

```
case
        x=y     then S1
        z>y+3   then S2
        …       then
endcase
```

Ricerca dicotomica



Altra forma di non determinismo solitamente nascosta

Algoritmi di ricerca

- Gli algoritmi di ricerca sono sostanzialmente una “simulazione” di algoritmi non deterministici
  - L’elemento cercato sta nella radice?

    Se sì, bene

    Altrimenti

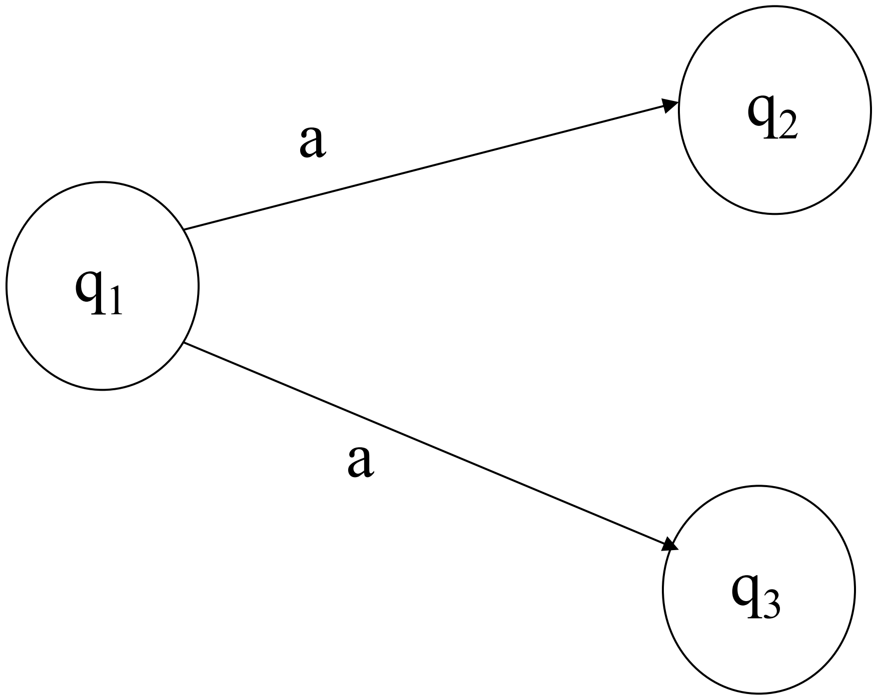
      Cerca nel sottoalbero sinistro

      Cerca nel sottoalbero destro
- La scelta della priorità tra cammini è spesso arbitraria

In conclusione

- Il non determinismo (ND) è un modello di computazione o quantomeno un modello di computazione parallela
  - Ada e altri linguaggi concorrenti lo sfruttano
- E’ un’astrazione utile per descrivere problemi e algoritmi di ricerca
- Può essere applicata a vari modelli computazionali
- Importante: i modelli ND non vanno confusi con i modelli stocastici

Aggiunta del non determinismo



$$\delta(q_1, a) = \{q_2, q_3\}$$

FSA non deterministici

- Un FSA non deterministico (NFA) è una tupla

  $\langle Q, I, \delta, q_0, F \rangle$, dove

  – Q, I, q_0, F sono definiti come in un FSA deterministico (DFA)

  – $\delta: Q \times I \rightarrow \mathcal{P}(Q)$

- Che cosa succede a δ^*?

Sequenza di mosse

- δ^* è definito induttivamente da δ

$$\delta^*(q, \varepsilon) = \{q\}$$

$$\delta^*(q, y.i) = \bigcup_{q' \in \delta^*(q,y)} \delta(q', i)$$

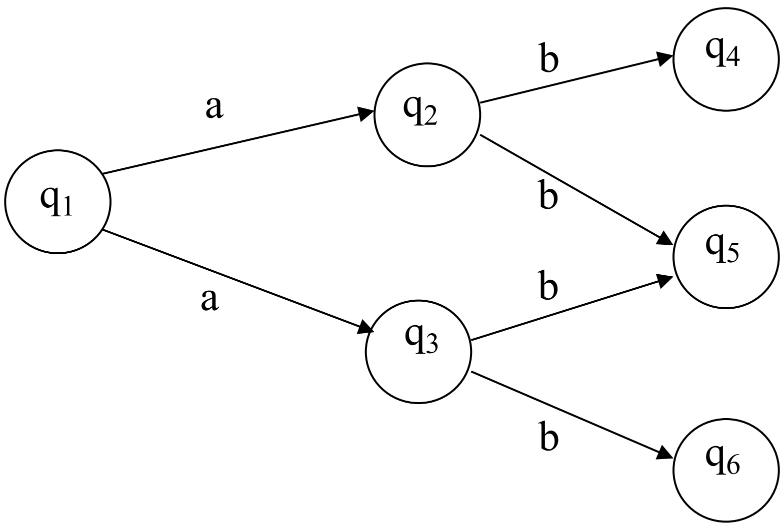
- Esempio:

$\delta(q_1,a) = \{q_2, q_3\}$,

$\delta(q_2,b) = \{q_4, q_5\}$,

$\delta(q_3,b) =\{q_6, q_5\}$

$\rightarrow \delta^*(q_1,ab) = \{q_4, q_5, q_6\}$



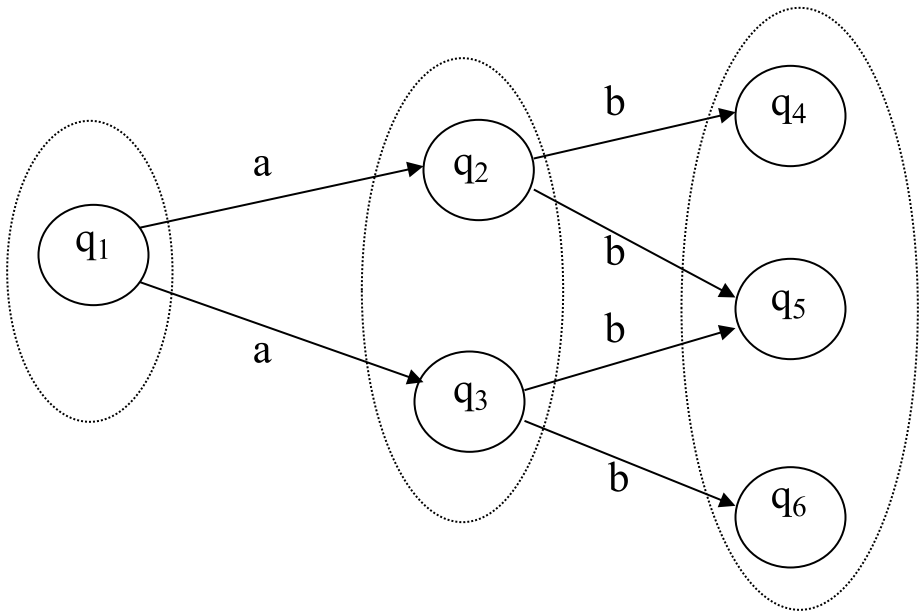
Condizione di accettazione

$$x \in L \Leftrightarrow \delta^*(q_0,x) \cap F \neq \emptyset$$

Tra le varie possibili esecuzioni (con lo stesso ingresso) dell'NFA, è sufficiente che una di esse vada a buon fine per accettare la stringa di ingresso

- Non determinismo <u>esistenziale</u>
  - C'è anche un'interpretazione universale: $\delta^*(q_0,x) \subseteq F$

DFA e NFA



- A partire da q_1 e leggendo ab l'automa raggiunge uno stato che appartiene all'insieme $\{q_4, q_5, q_6\}$
- Chiamiamo ancora "stato" l'insieme di possibili stati in cui l'NFA può trovarsi durante l'esecuzione

Formalmente

- NFA e DFA hanno lo stesso potere espressivo
- Dato un NFA, si può sintetizzare <u>automaticamente</u> un DFA come segue

  Se $A_{ND} = \langle Q, I, \delta, q_0, F \rangle$ allora

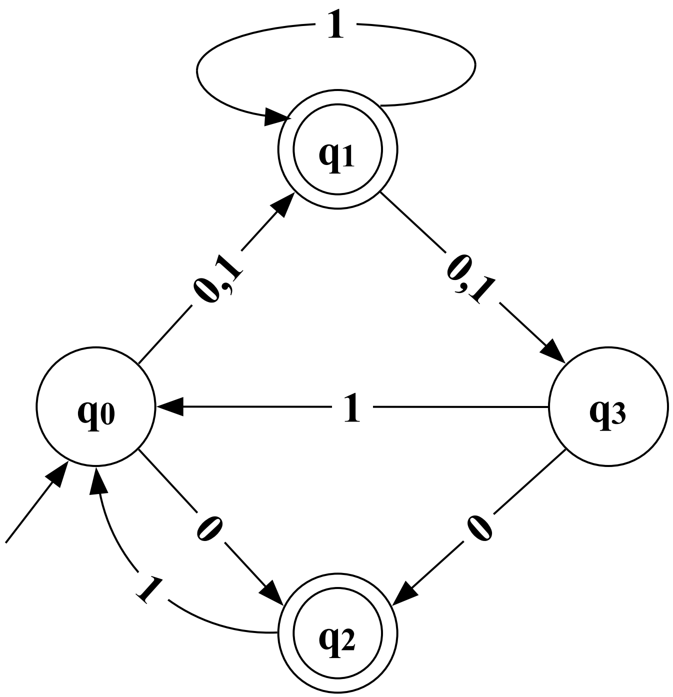
  $A_D = \langle Q_D, I, \delta_D, q_{0D}, F_D \rangle$, dove

  - $Q_D = \mathcal{P}(Q)$
  - $$\delta_D(q_D, i) = \bigcup_{q \in q_D} \delta(q, i)$$
  - $q_{0D} = \{q_0\}$
  - $F_D = \{q_D \mid q_D \in Q_D \wedge q_D \cap F \neq \emptyset\}$

Esempio (1)

Trasformare il seguente NFA nel DFA equivalente

q0	0	q1
q0	0	q2
q0	1	q1
q1	0	q3
q1	1	q3
q1	1	q1
q2	0	⊥
q2	1	q0
q3	0	q2
q3	1	q0



Esempio (2)

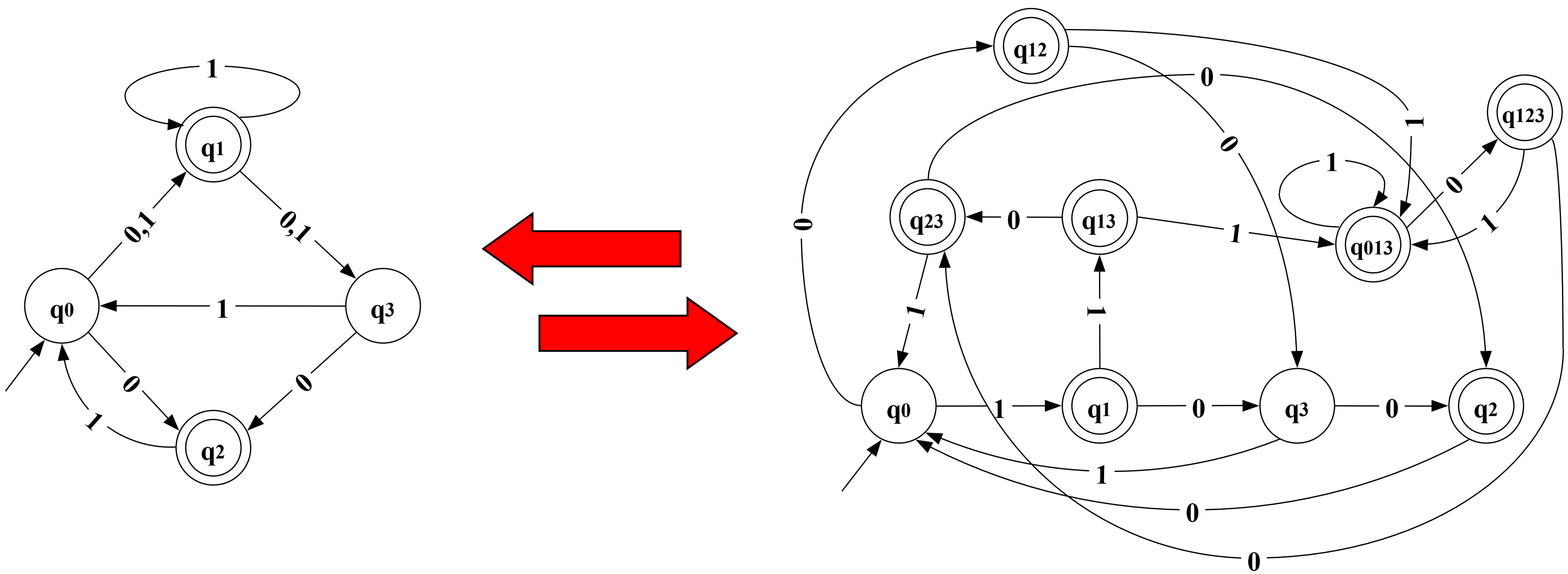
- Procediamo ricorsivamente

q0	0	q1
q0	0	q2
q0	1	q1
q1	0	q3
q1	1	q3
q1	1	q1
q2	0	⊥
q2	1	q0
q3	0	q2
q3	1	q0

q0	0	q12
q0	1	q1
q1	0	q3
q1	1	q13
q2	0	⊥
q2	1	q0
q3	0	q2
q3	1	q0
q12	0	q3
q12	1	q013
q13	0	q23
q13	1	q013
q013	0	q123
q013	1	q013
q23	0	q2
q23	1	q0
q123	0	q23
q123	1	q013

Esempio (3)

Graficamente



Perché il ND?

- Gli NFA non sono più potenti dei DFA, ma non sono inutili
  - Può essere più semplice progettare un NFA
  - Possono essere esponenzialmente più piccoli per quanto riguarda il numero di stati
- Esempio: un NFA con 5 stati diventa nel peggiore dei casi un FSA con 2^5 stati

TM non deterministiche

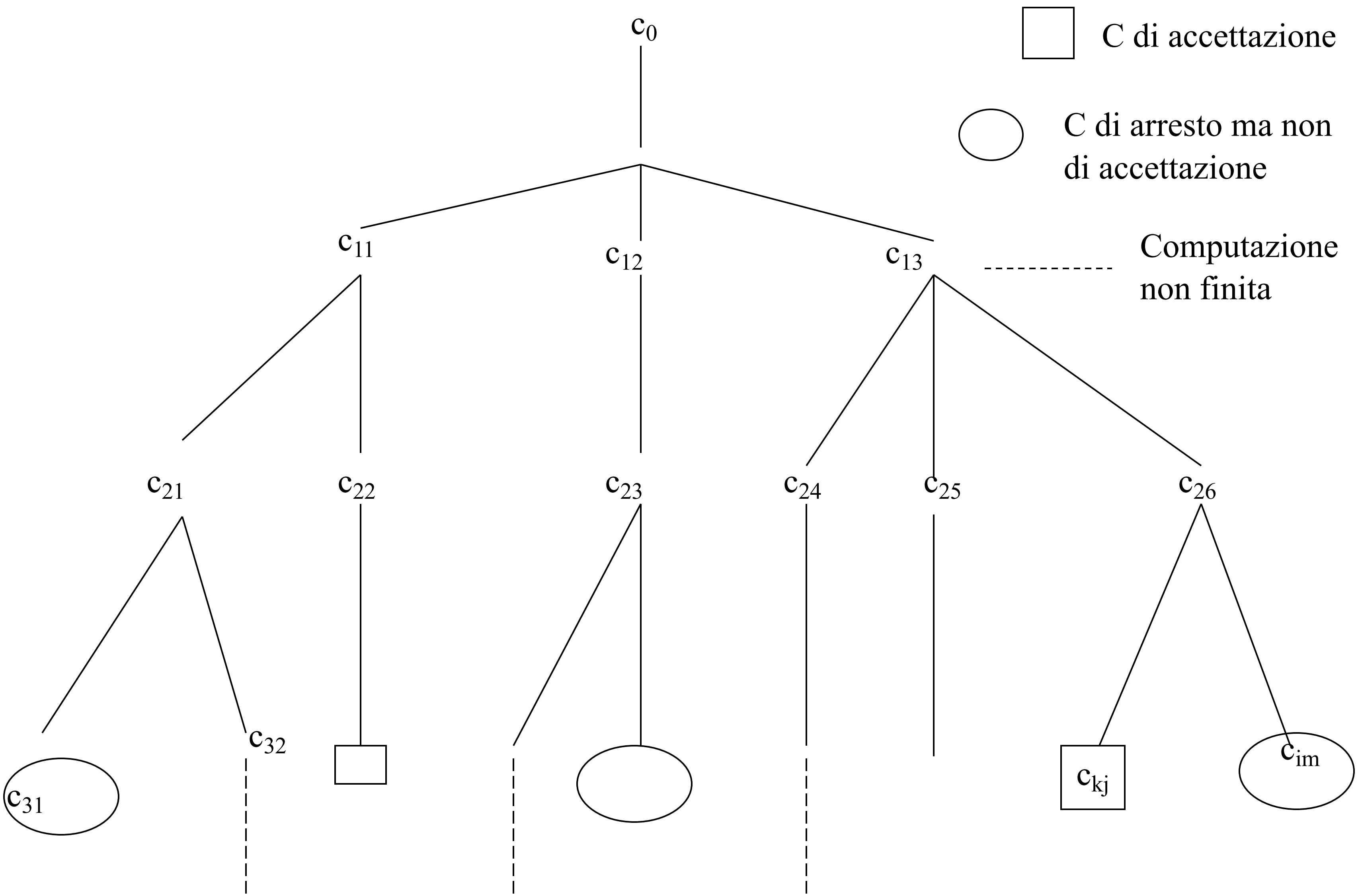
- Per definire una TM non deterministica (NTM), occorre cambiare la funzione di transizione e la funzione di traduzione
- Tutti gli altri elementi sono come nella (D)TM
- La funzione di transizione è

$$\delta\text{: (Q-F) x I x } \Gamma^k \rightarrow \mathcal{P}(\text{Q x } \Gamma^k \text{ x } \{R,L,S\}^{k+1})$$

e la funzione di uscita

$$\eta\text{: (Q-F) x I x } \Gamma^k \rightarrow \mathcal{P}(\text{O x } \{R,S\})$$

Albero di computazione



Condizione di accettazione

- Una stringa $x \in I^*$ è accettata da una NTM se e solo se <u>esiste una computazione</u> che termina in uno stato di accettazione
- Sembrerebbe che il problema di accettare una stringa si possa ridurre alla visita di un albero di computazione
  - Come dovremmo eseguire la visita?
  - Che rapporto c'è tra DTM e NTM?

Visita dell'albero di computazione

- Esistono diverse modalità di visita:
  - In profondità (Depth-first)
  - In ampiezza (Breadth-first)
- Una visita in profondità non può funzionare per il nostro problema perché l'albero di computazione potrebbe avere un cammino infinito e l'algoritmo potrebbe "bloccarsi" in esso
- Dovremmo adottare un algoritmo di visita in ampiezza

DTM e NTM

Possiamo costruire una DTM che visita un albero livello dopo livello?

E' un esercizio lungo (e noioso), ma <u>si può fare</u>

- Possiamo costruire una DTM che stabilisce se una NTM riconosce una stringa x
- Data una NTM, possiamo costruire una DTM equivalente

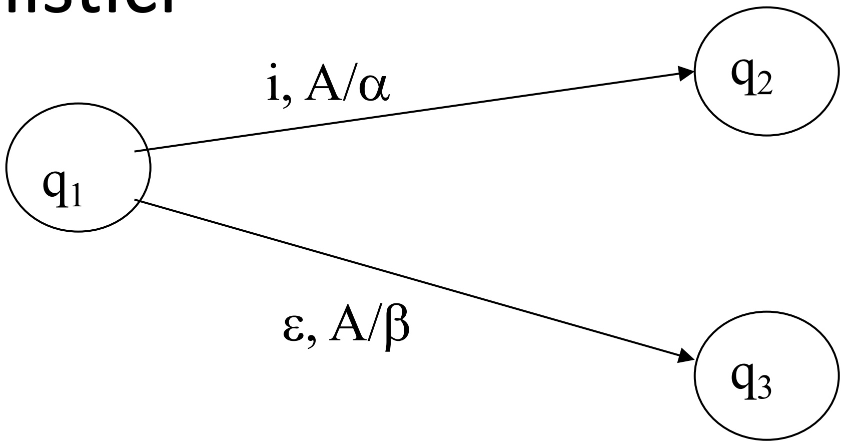
Il ND non aggiunge potere espressivo alle TM

ε-mosse e PDA

- Le ε-mosse avevano il seguente vincolo:

$$\text{Se } \delta(q,\varepsilon,A) \neq \perp, \text{ allora } \delta(q,i,A) = \perp \quad \forall i \in I$$

- Senza questo vincolo, la presenza di ε-mosse renderebbe i PDA intrinsecamente non deterministici



Aggiunta del non determinismo ai PDA

- Rimuovere il vincolo rende già i PDA non deterministici
- Inoltre possiamo avere non determinismo cambiando la funzione di transizione di un PDA e di conseguenza:
  - Le transizioni tra configurazioni
  - La condizione di accettazione

Definizione

Un PDA non deterministico (NPDA) è una tupla $<Q, I, \Gamma, \delta, q_0, Z_0, F>$

- $Q, I, \Gamma, q_0, Z_0, F$ come nel (D)PDA
- δ è la funzione di transizione definita come

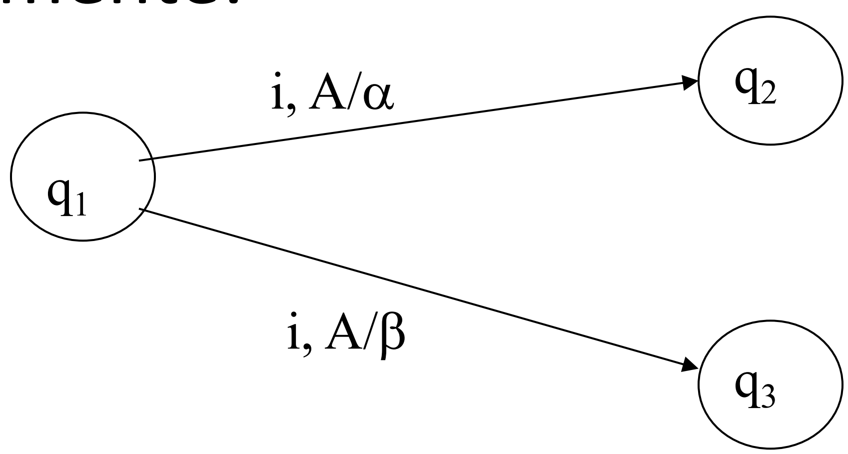
$$\delta: Q \times (I \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_{\mathcal{F}}(Q \times \Gamma^*)$$

- Cos'è la $\mathcal{F}$ in $\mathcal{P}_{\mathcal{F}}$?
- Perché $\mathcal{F}$?

Funzione di transizione

$$\delta \colon Q \times (I \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_{\mathcal{F}}(Q \times \Gamma^*)$$

- $\mathcal{P}_{\mathcal{F}}$ indica i sottoinsiemi *finiti* di $Q \times \Gamma^*$
  - Perché non l'abbiamo indicato per le NTM?
- Graficamente:



Transizione tra configurazioni

La relazione $\vdash$ su $Q \times I^* \times \Gamma^* \times Q \times I^* \times \Gamma^*$ è definita da $c=\langle q,x,\gamma \rangle \vdash c'=\langle q', x', \gamma' \rangle$ se e solo se

- Caso 1
  - $x=iy$, $x'=y$
  - $\gamma=A\beta$, $\gamma'=\alpha\beta$
  - $\langle q', \alpha \rangle \in \delta(q,i,A)$
- Caso 2
  - $x'=x$
  - $\gamma=A\beta$, $\gamma'=\alpha\beta$
  - $\langle q', \alpha \rangle \in \delta(q,\varepsilon,A)$

Condizione di accettazione

- Dato un NPDA P

$\forall x \in I^* \; (x \in L(P) \Leftrightarrow \exists q \exists \gamma \; c_0 = <q_0, x, Z_0> \vdash^* c_F = <q, \varepsilon, \gamma>$ e $q \in F)$

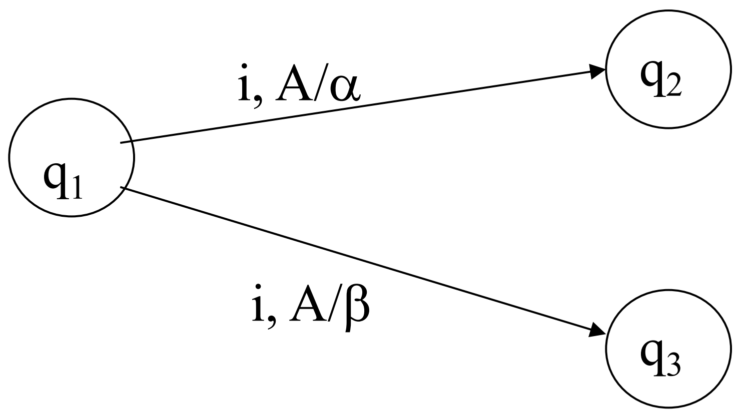
- Informalmente, una stringa è accettata da un PDA se <u>c'è un cammino</u> coerente con x sul PDA che va dallo stato iniziale a uno stato finale
  - La stringa di ingresso dev'essere letta interamente

Effetti del non determinismo

- Il ND non aggiunge potere espressivo a
  - TM
  - FSA
- Ne aggiunge ai DPDA?

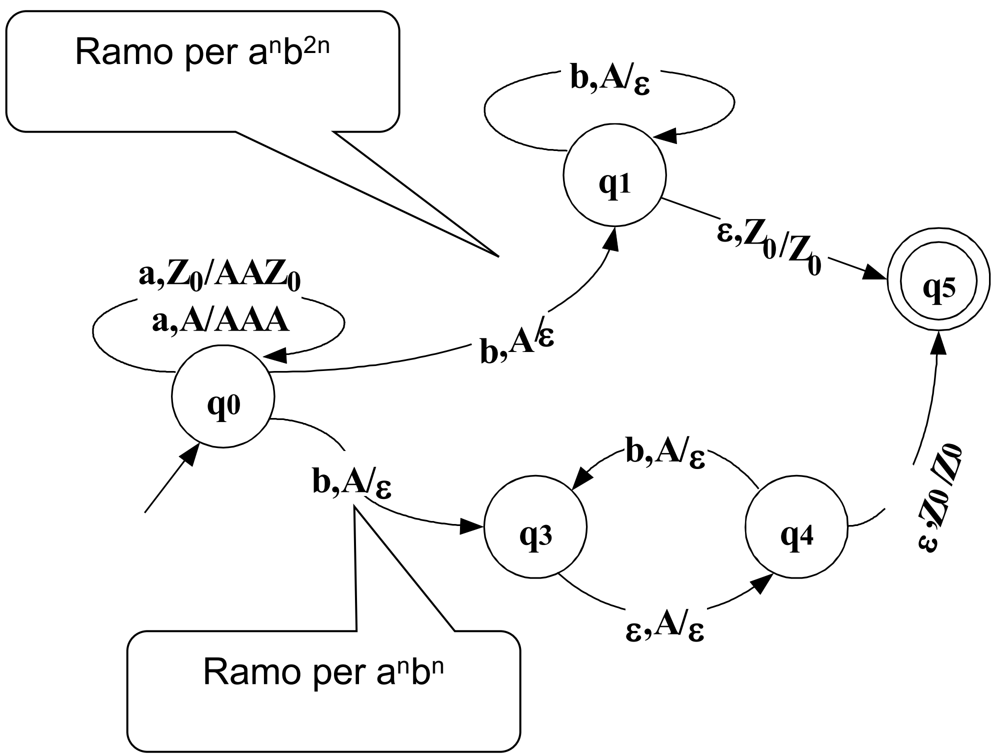
NPDA e DPDA (1)

- Ovviamente un NPDA può riconoscere tutti i linguaggi riconoscibili con un DPDA
- Il ND consente

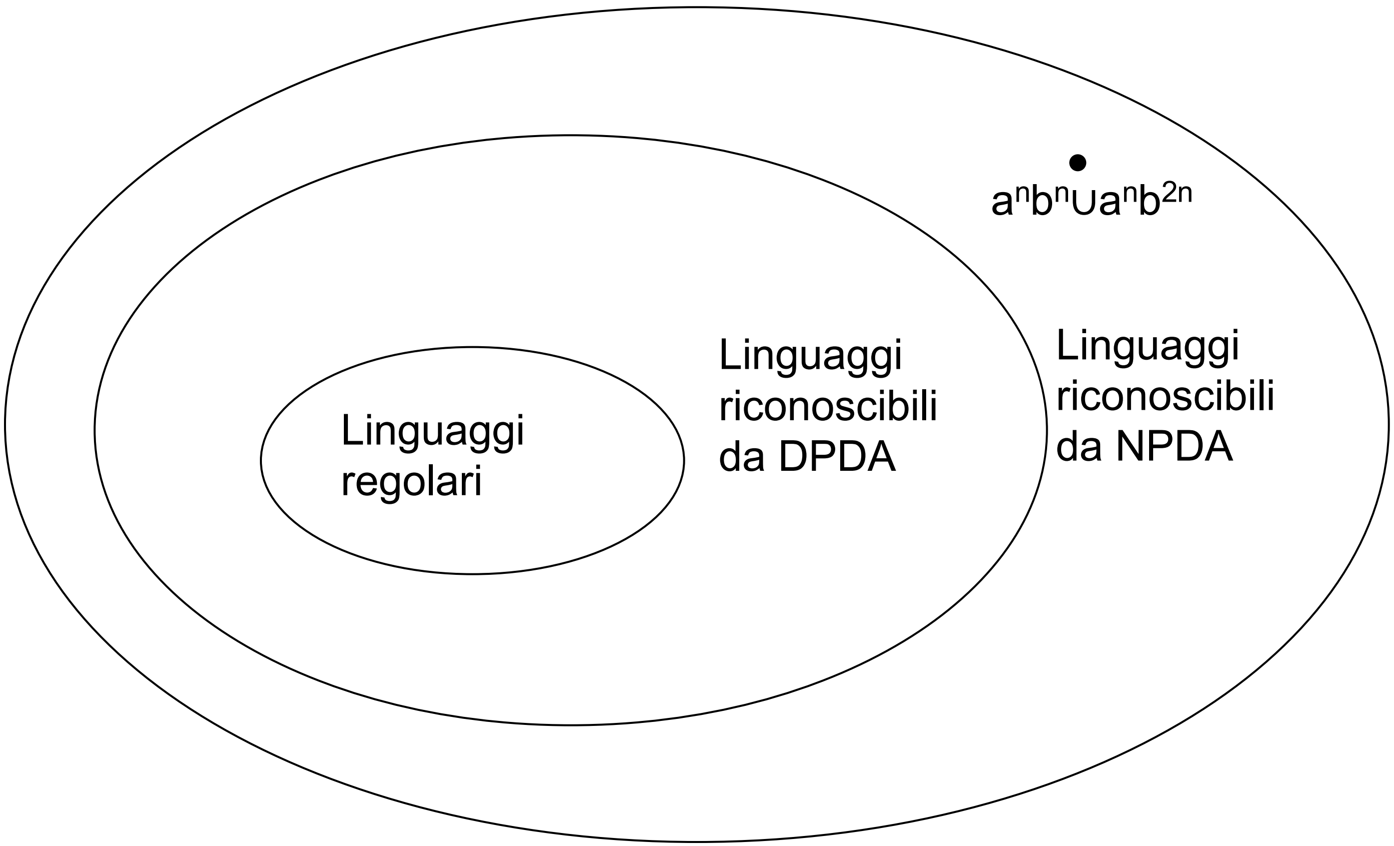


quindi gli NPDA possono riconoscere $\{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$

$\{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$

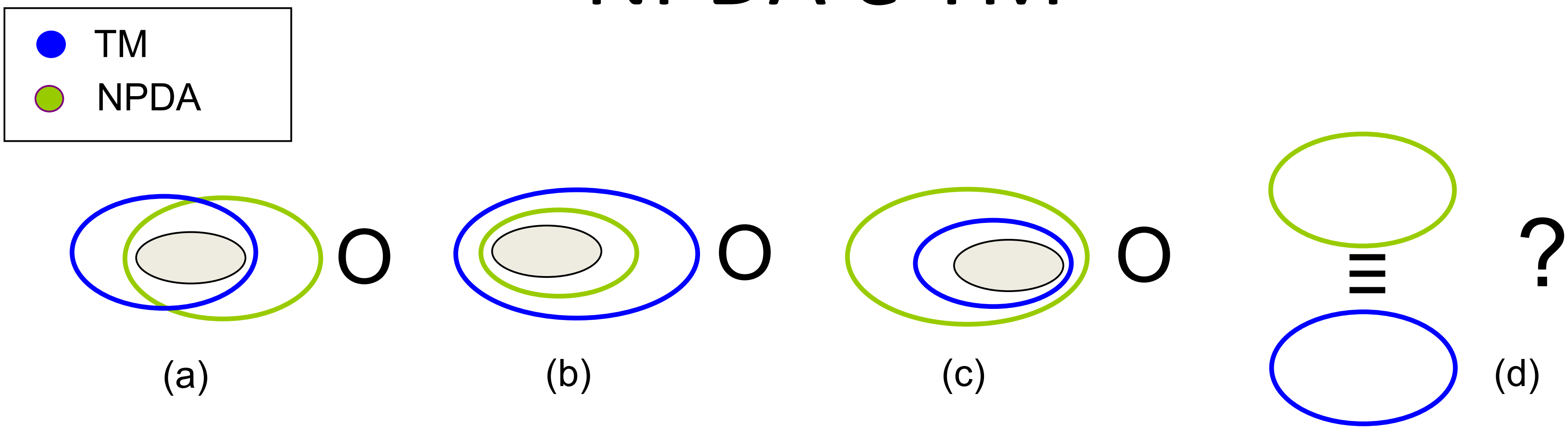


NPDA e DPDA (2)



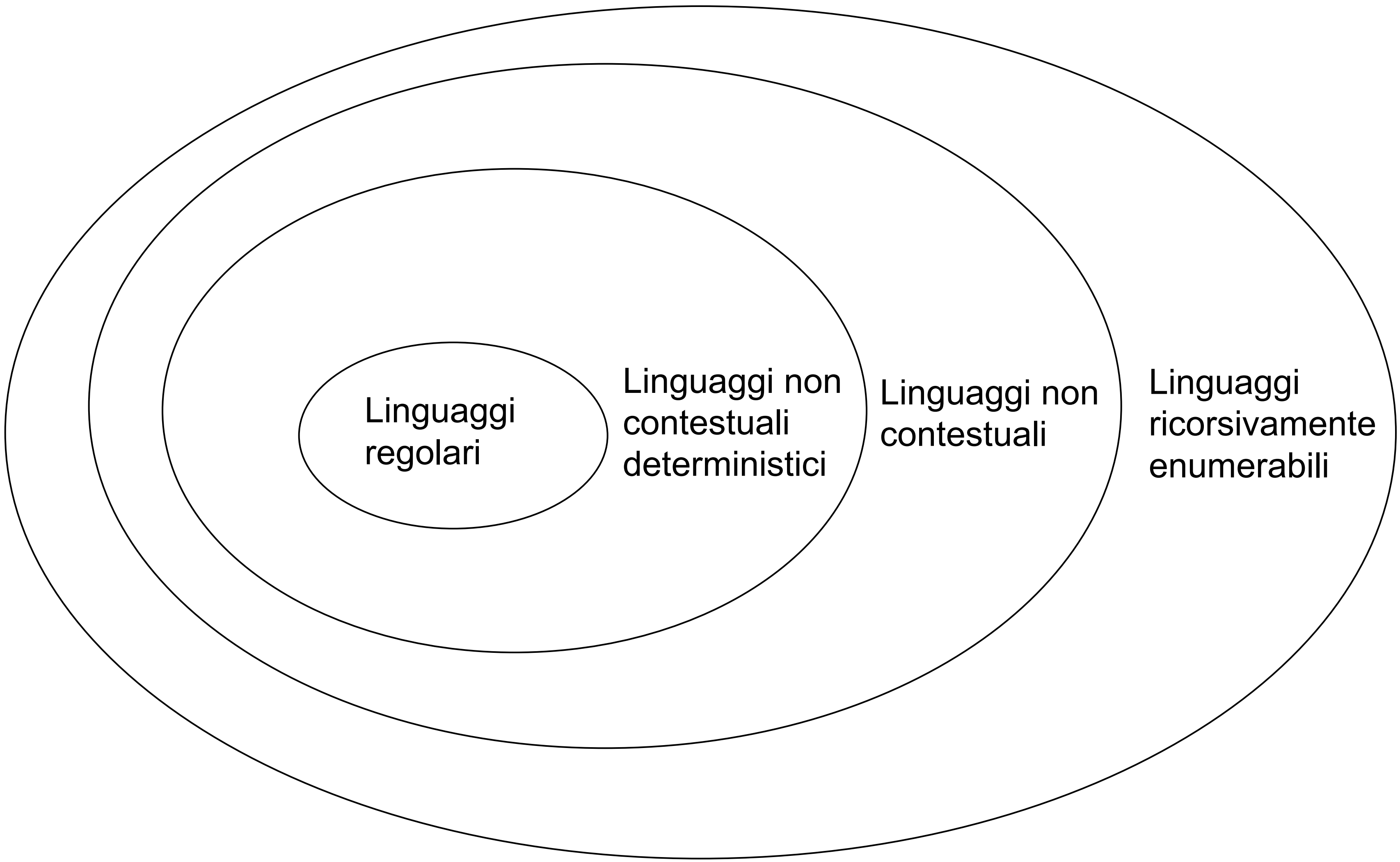
I linguaggi riconoscibili da NPDA si chiamano linguaggi non contestuali (<u>context-free</u>)

NPDA e TM



- (a) e (c): NO!
  - Una (N)TM può simulare un NPDA usando il nastro come una pila
- (d): NO!
  - La pila è ancora una memoria distruttiva
  - $a^n b^n c^n$ non è riconoscibile da un NPDA

Il bersaglio

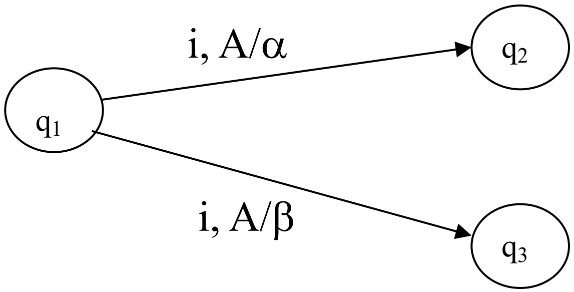


Proprietà di chiusura nei DPDA

- Nei DPDA abbiamo
  - Chiusura rispetto al complemento
  - Non chiusura rispetto a unione, intersezione e differenza
- Cambiare il potere espressivo dell'automa può cambiarne il comportamento rispetto alle operazioni!

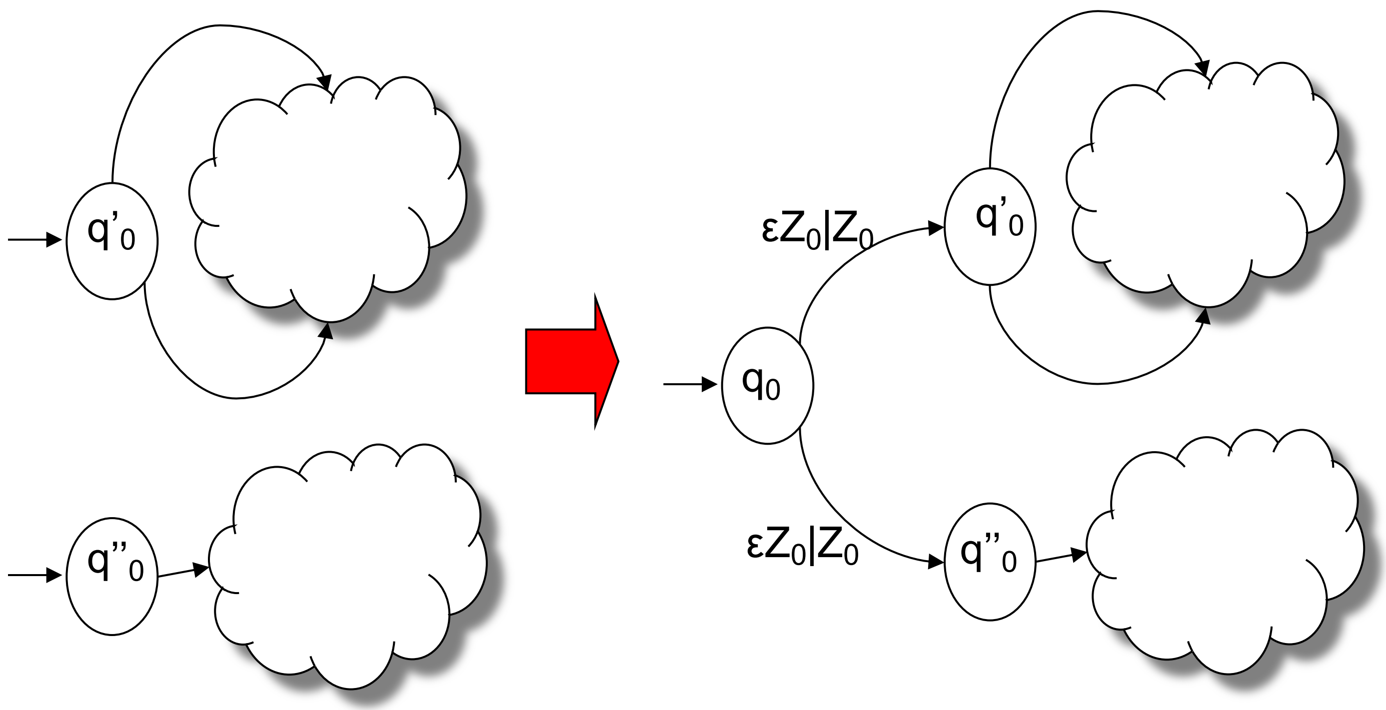
Unione (1)

- Gli NPDA sono <u>chiusi</u> rispetto all'unione
  - Intuizione:



- Dati due NPDA, P_1 e P_2, possiamo sempre costruire un NPDA che rappresenti l'unione creando un nuovo stato iniziale che è connesso a entrambi gli stati iniziali di P_1 e P_2 con una ε-mossa

Unione (2)



Intersezione

- La chiusura rispetto all'intersezione continua a non valere
- Consideriamo
  - $\{a^n b^n c^*\}$
  - $\{a^* b^n c^n\}$

  Entrambi sono riconoscibili mediante (N)PDA, ma

  $\{a^n b^n c^*\} \cap \{a^* b^n c^n\} = \{a^n b^n c^n\}$ non è riconoscibile da alcun NPDA

Complemento (1)

- Se una classe di linguaggi è chiusa rispetto all’unione ma non rispetto all’intersezione, non può essere chiusa rispetto al complemento
  - Possiamo scrivere l’intersezione in termini di unione e complemento
- Gli NPDA <u>non sono chiusi</u> rispetto al complemento

Note

- Se una macchina è deterministica e la sua computazione termina, il complemento si può ottenere
  - Completando la macchina
  - Scambiando gli stati di accettazione con quelli di non accettazione
- Il non determinismo, come la non terminazione (computazione infinita) rende questo approccio inapplicabile

Complemento (2)

- Per gli NPDA, le computazioni possono sempre essere fatte terminare (come per i DPDA)
- Tuttavia, il ND può causare questo problema:

  Si possono avere due computazioni:

  - $c_0 = \langle q_0, x, Z_0 \rangle \vdash^* c_1 = \langle q_1, \varepsilon, \gamma \rangle$
  - $c_0 = \langle q_0, x, Z_0 \rangle \vdash^* c_2 = \langle q_2, \varepsilon, \gamma \rangle$

  con $q_1 \in F$ e $q_2 \notin F$

→ anche se scambiamo gli stati di accettazione e non accettazione, x è comunque accettato!