Il riconoscimento dei linguaggi liberi

Prof. A. Morzenti

aa 2008-2009

RICONOSCIMENTO DEI LINGUAGGI LIBERI

AUTOMA PER LINGUAGGI LIBERI:
- con stati e memoria a pila
- mosse dell'automa in corrisp. con le regole della grammatica
- non sempre si può ottenere un automa deterministico
- la presenza di due memorie (stati e pila) complica il quadro

Vedremo:
- automi a pila
- automi deterministici (per linguaggi liberi appartenenti a DET)
- algoritmi di parsificazione deterministici (veloci, in t. lineare / per LL(k) e LR(k))
- un algoritmo generale (Earley)

AUTOMI A PILA

$$\overbrace{A_1 A_2 \ldots A_k}^{}\quad \text{(cima: } A_k\text{)}$$

$$a_1 a_2 \ldots \underbrace{}_{} \overset{car.corrente}{a_i} \ldots a_n \dashv$$

1) memoria ausiliaria, organizzata come una pila illimitata di simboli
2) stringa di ingresso (o sorgente)
3) operazioni applicabili:
   - <u>impilamento</u>: *push(B), push(*B_1*,* B_2*, ...* B_n*)*: pone il/i simbolo/i in cima (a dx di A_k)
   - <u>test di pila vuota</u>: *empty* predicato vero solo se k = 0
   - <u>spilamento</u>: *pop,* se la pila non è vuota, toglie dalla cima A_k
4) Z_0 simbolo detto *il fondo* (può solo essere letto)
5) -| *carattere terminatore* della stringa di ingresso
6) configurazione: stato corrente, car. corrente, contenuto della pila

UNA MOSSA DELL'AUTOMA:
- legge car. corrente avanzando con la testina o compie mossa spontanea senza muovere la testina
- legge il simbolo in cima e lo toglie dalla pila o legge Z_0 se la pila è vuota
- in base a car. corrente, stato corr, e simbolo letto dalla pila, calcola il nuovo stato ed esegue l'eventuale impilamento di uno o più simboli

DEFINIZIONE DELL'AUTOMA A PILA

Un automa a pila M (in generale non deterministico) è definito da:

1. *Q* l'*inisieme* finito degli *stati* dell'unità di controllo
2. *Σ* *l'alfabeto di ingresso*
3. *Γ* *l'alfabeto della pila*
4. δ la *funzione di transizione*
5. $q_0 \in Q$ lo *stato iniziale*
6. $Z_0 \in \Gamma$ il *simbolo iniziale della pila*
7. $F \subseteq Q$ *l'insieme degli stati finali*

mossa spontanea

FUNZIONE DI TRANSIZIONE:

dominio: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$

codominio: le ***parti finite*** dell'insieme $Q \times \Gamma^*$

indeterminismo

MOSSA CON LETTURA:
l'automa, nello stato *q* con Z in cima alla pila, leggendo *a* può entrare in uno degli stati p_i $1 \leq i \leq n$, dopo aver eseguito in succ. le operazioni pop, push(γ_i).

$$\boxed{\begin{array}{l}\delta(q,a,Z)=\{(p_1,\gamma_1),(p_2,\gamma_2),\ldots(p_n,\gamma_n)\} \\ \text{con } n \geq 1, a \in \Sigma, Z \in \Gamma \text{ e con } p_i \in Q, \gamma_i \in \Gamma^*\end{array}}$$

NOTA: la scelta dell'azione *i*-esima tra le *n* possibili non è deterministica; l'avanzamento è automatico; il simbolo in cima è sempre spilato; la stringa impilata può essere vuota.

MOSSA SPONTANEA:
l'automa, nello stato *q* con Z in cima alla pila, senza leggere alcun carattere di ingresso, può entrare in uno degli stati p_i, $1 \leq i \leq n$, dopo aver eseguito in succ. le operazioni pop, push(γ_i).

$$\boxed{\begin{array}{l}\delta(q,\varepsilon,Z)=\{(p_1,\gamma_1),(p_2,\gamma_2),\ldots(p_n,\gamma_n)\} \\ \text{con } n \geq 1, Z \in \Gamma, p_i \in Q, \gamma_i \in \Gamma^*\end{array}}$$

NON DETERMINISMO: per una data terna (stato, cima-pila, ingresso) si hanno due o più possibilità tra mosse che consumano ingresso e mosse spontanee

CONFIGURAZIONE ISTANTANEA DELLA MACCHINA M: una terna

$$\boxed{(q, y, \eta) \in Q \times \Sigma^* \times \Gamma^+}$$

che descrive:

- q	lo stato attuale del controllo
- *y*	la parte ancora da leggere della stringa in ingresso *x*
- *η*	la stringa contenuta in pila

CONFIGURAZIONE INIZIALE: (q_0, x, Z_0)

Una CONFIGURAZIONE (q, ε, η) o $(q, -|, \eta)$ è <u>FINALE</u> se $q \in F$

(NB: ingresso consumato tutto)

<u>TRANSIZIONE DA UNA CONFIGURAZIONE A UN'ALTRA</u>: $\boxed{(q, y, \eta) \to (p, z, \lambda)}$

CATENA DI PIÙ TRANSIZIONI indicata con: $\stackrel{+}{\to}$

Conf. precedente	Conf. successiva	Mossa applicata
$(q, az, \eta Z)$	$(p, z, \eta\gamma)$	mossa con lettura $\delta(q,a,Z)=\{(p,\gamma),...\}$
$(q, az, \eta Z)$	$(p, az, \eta\gamma)$	mossa spontanea $\delta(q,\varepsilon,Z)=\{(p,\gamma),...\}$

NB: Ogni mossa cancella il simbolo in cima, ma esso può essere impilato di nuovo dalla stessa mossa, se si intende mantenerlo in pila, perchè si impila una ***stringa***.

Una stringa x è riconosciuta (o accettata) dalla macchina M mediante stato finale se:

$$\boxed{(q_0, x, Z_0) \xrightarrow{+} (q, \varepsilon, \lambda)}$$

q∈ F e λ ∈ Γ* (nessuna condizione su λ, solo in alcuni casi può essere la stringa vuota)

DIAGRAMMA STATO-TRANSIZIONE PER AUTOMI A PILA

ESEMPIO: palindromi di lunghezza pari, accettate mediante stato finale dal riconoscitore a pila.

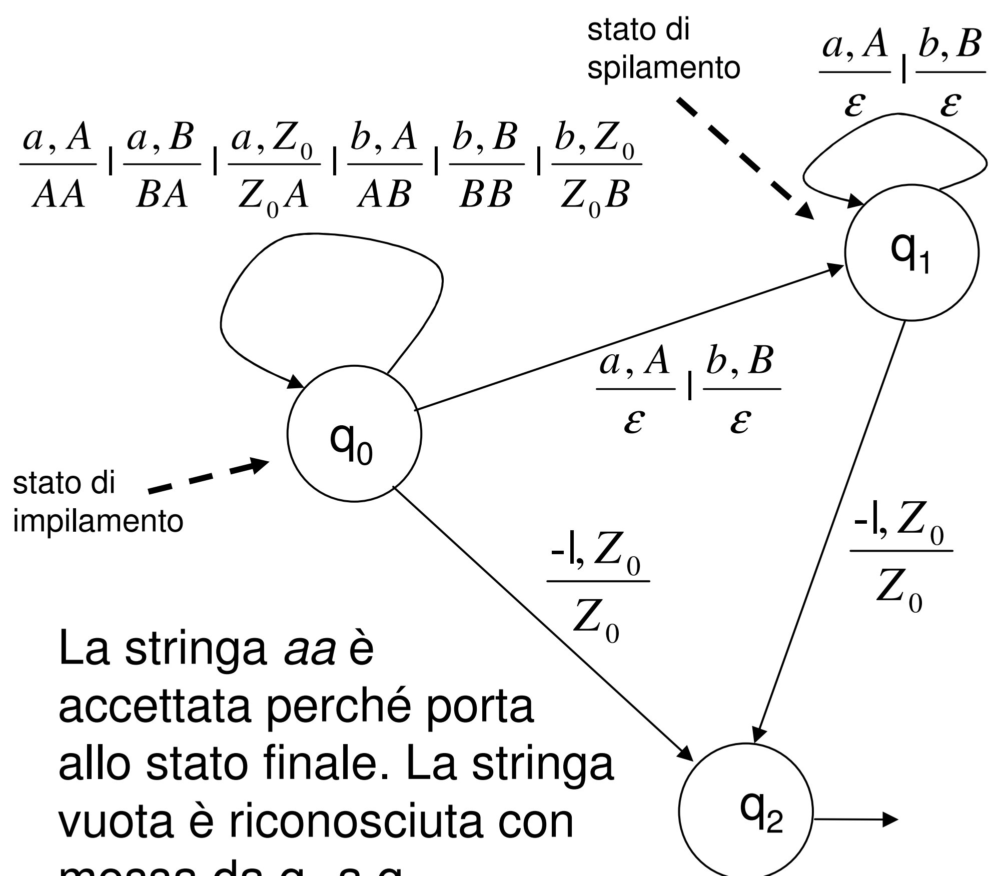
$$L = \left\{ uu^R \mid u \in \{a,b\}^* \right\}$$

Pila	x	Stato	Commento
Z_0	aa-|	q_0	
Z_0A	a-|	q_0	
Z_0AA	-|	q_0	rifiuto: nessuna mossa definita per (q_0, -|, A)

"scommette" che |x|>2

Pila	x	Stato	Commento
Z_0	aa-|	q_0	
Z_0A	a-|	q_0	
Z_0	-|	q_1	
Z_0	ε	q_2	riconoscimento nello stato finale

"scommette" che |x|=2



La stringa *aa* è accettata perché porta allo stato finale. La stringa vuota è riconosciuta con mossa da q_0 a q_2.

DALLA GRAMMATICA ALL'AUTOMA A PILA

1) Le regole della grammatica possono essere viste come istruzioni di una macchina a pila ***non deterministica*** che riconosce il linguaggio (essa non usa gli stati come memoria ma solo la pila).Intuitivamente: l'automa opera in modo *predittivo* (goal oriented) e usa la pila come agenda delle future azioni da compiere.

2) I simboli della pila sono caratteri terminali e non terminali. Se la pila contiene A_1, ... A_k, la macchina esegue prima l'operazione associata a A_k, che ha l'obiettivo di riconoscere se in ingresso, a partire dal carattere corrente a_i, vi sia una stringa *w* derivabile da A_k; in caso positivo l'azione fa avanzare la testina di |w| posizioni

3) L'obiettivo può articolarsi ***ricorsivamente*** in sotto-obiettivi, se per riconoscere A_k è necessario riconoscere altri simboli terminali o non

Inizialmente l'obiettivo è l'assioma della grammatica: compito del riconoscitore è infatti riconoscere se la stringa sorgente derivi dall'assioma. Inizialmente la pila contiene solo il simbolo di fondo Z_0 e l'assioma S e la testina di lettura è posta sul primo carattere della stringa sorgente. A ogni passo l'automa sceglie (indeterministicamente) una delle regole applicabili ed esegue la mossa corrispondente. L'automa riconosce la stringa se alla lettura del term. -| la pila è vuota.

DATA LA GRAMMATICA $G=(V, \Sigma, P, S)$, $A,B \in V$, $b \in \Sigma$, $A_i \in V \cup \Sigma$

Regola	Mossa	Commento
$A \to BA_1 \ldots A_n \quad n \geq 0$	**if** *cima = A* **then** pop; push(A_n ... A_1B) **end if**	per riconoscere A si devono riconoscere B A_1 ... A_n
$A \to bA_1 \ldots A_n \quad n \geq 0$	**if** *car-corr = b Λ cima = A* **then** pop; push(A_n ... A_1); avanza testina lett **end if**	*b* era il primo carattere atteso ed è stato letto; restano da riconoscere A_1 ... A_n
$A \to \varepsilon$	**if** *cima = A* **then** pop **end if**	È stata riconosciuta ε che deriva da A
Per ogni car. $b \in \Sigma$	**if** *car-corr = b Λ cima = b* **then** pop; avanza testina lettura **end if**	*b* era il primo carattere atteso ed è stato letto
– – –	**if** *car-corr = -| Λ pila è vuota* **then** accetta **end if** alt	stringa tutta scandita, non restano in agenda altri obiettivi

ESEMPIO – Regole e mosse del riconoscitore predittivo del linguaggio

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

Regola	Mossa
1. $S \to aS$	**if** *car-corr = a Λcima=S* **then** pop; push(S); avanza **end if** ($\delta(q_0,a,S)=(q_0,S)$)
2. $S \to A$	**if** *cima = S* **then** pop; push(A) **end if** ($\delta(q_0,\varepsilon,S)=(q_0,A)$)
3. $A \to aAb$	**if** *car-corr = a Λcima = A* **then** pop; push(bA); avanza **end if** ($\delta(q_0,a,A)=(q_0,bA)$)
4. $A \to ab$	**if** *car-corr = a Λcima = A* **then** pop; push(b); avanza **end if** ($\delta(q_0,a,A)=(q_0,b)$)
5.	**if** *car-corr = b Λcima = b* **then** pop; avanza **end if** ($\delta(q_0,b,b)=(q_0,\varepsilon)$)
6.	**if** *car-corr = -| Λpila vuota* **then** accetta **end if** alt

Nondeterminismo tra mosse 1 e 2 (2 può essere scelta anche quando c'è *a* in ingresso), e tra 3 e 4.

Stringa $a^n b^m$, $n \geq m \geq 1$ analizzata come $a^{n-m} a^m b^m$, "indovinando nondeterministicamente" il punto in cui inizia $a^m b^m$ scegliendo tra mosse 1 e 2. Inoltre "indovina" il punto in cui finisce a^m e inizia b^m scegliendo tra mosse 3 e 4

L'automa così costruito riconosce una stringa se, e solo se, la grammatica la genera: per ogni calcolo che termina con successo esiste una derivazione corrispondente e viceversa. L'automa simula le derivazioni sx della grammatica (perchè i caratteri in ingresso sono solo simboli terminali).

$S \Rightarrow A \Rightarrow aAb \Rightarrow aabb$

traccia con esito positivo:

Pila	x
Z_0S	$aabb$-|
$\delta(q_0,\varepsilon,S)=(q_0,A)$	
Z_0A	$aabb$-|
$\delta(q_0,a,A)=(q_0,bA)$	
Z_0bA	abb-|
$\delta(q_0,a,A)=(q_0,b)$	
Z_0bb	bb-|
$\delta(q_0,b,b)=(q_0,\varepsilon)$	
Z_0b	b-|
$\delta(q_0,b,b)=(q_0,\varepsilon)$	
Z_0	-|

Ma l'algoritmo non sa quale sarà la derivazione giusta: deve esplorare tutte le possibilità, anche quelle che falliranno

$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaA \Rightarrow errore$

$S \Rightarrow aS \Rightarrow aA \Rightarrow aaAb \Rightarrow errore$

$S \Rightarrow aS \Rightarrow aA \Rightarrow aab \Rightarrow errore$

$S \Rightarrow A \Rightarrow ab \Rightarrow errore$

Una stringa è accettata da diverse computazioni se, e solo se, è ambigua per la grammatica.

Le regole della tabella precedente sono bidirezionali: possono essere applicate in senso inverso per generare la grammatica partendo dall'automa a pila.

Mettendo insieme la trasformazione diretta e inversa si ottiene:

PROPRIETÀ – La famiglia dei linguaggi liberi coincide con quella dei linguaggi riconosciuti a pila vuota da un automa a pila indeterministico, avente un solo stato.

PURTROPPO L'AUTOMA NON È DETERMINISTICO ED EPLORA TUTTE LE VIE CON COMPLESSITÀ DI CALCOLO NON POLINOMIALE RISPETTO ALLA LUNGHEZZA DELLA STRINGA SORGENTE ... vedremo algoritmi più efficienti ...

VARIETÀ DI AUTOMI A PILA

L'automa a pila definito nei casi pratici si differenzia da quello precedentemente costruito in modo diretto dalla grammatica, in due modi: ha gli stati interni e usa una diversa condizione per accettare le stringhe.

MODALITÀ DI ACCETTAZIONE:

- riconoscimento a stato finale (prescinde dal contenuto della pila: la macchina entra in uno stato finale)

oppure

- riconoscimento a pila vuota (prescinde dallo stato in cui si trova l'automa)

oppure

- combinata: (a stato finale ***e*** a pila vuota)

PROPRIETÀ – Per la famiglia degli automi a pila indeterministici dotati di stati interni, le modalità di accettazione 1) a pila vuota, 2) a stato finale, 3) combinata (stato finale e pila vuota), hanno la stessa capacità di riconoscimento del linguaggio.

FUNZIONAMENTO SENZA CICLI SPONTANEI E IN LINEA

Se esiste ciclo di mosse spontanee automa potrebbe eseguire numero illimitato di mosse senza leggere alcun carattere d'ingresso

Ciò

- impedisce all'automa di leggere per intero la stringa sorgente;
- fa aumentare senza limite il tempo necessario per accettare decidere se accettare

MA: …

Si può sempre costruire un AUTOMA EQUIVALENTE PRIVO DI CICLI SPONTANEI.

UN AUTOMA FUNZIONA IN LINEA (on line) SE ESSO, ALLA LETTURA DELL'ULTIMO CARATTERE DELLA STRINGA, PUÒ SUBITO DECIDERE SE ACCETTARLA, SENZA FARE ULTERIORI MOSSE.

Un automa a pila può sempre essere convertito in un AUTOMA EQUIVALENTE DEL TIPO IN LINEA.

LINGUAGGI LIBERI E AUTOMI A PILA: UNA SOLA FAMIGLIA

Si può dimostrare che il linguaggio accettato da un automa a pila con stati è libero, e dato che abbiamo visto come ogni linguaggio libero sia riconosciuto da un automa a pila, vale l'enunciato:

PROPRIETÀ - La famiglia LIB dei linguaggi liberi coincide con quella dei linguaggi riconosciuti dagli automi a pila (non deterministici).

RICONOSCITORE DELL'INTERSEZIONE DI LINGUAGGI LIBERI E REGOLARI

L'affermazione che l'intersezione di un linguaggio libero con uno regolare è un linguaggio libero, è ora facile da giustificare.

Data la grammatica G e l'automa finito A, si mostra come ottenere l'automa a pila M che riconosce L(G) $\cap$ L(A):

1) costruiamo l'automa N che **a pila vuota** riconosce L(G)
2) costruiamo la machina M prodotto cartesiano delle due macchine N e A, applicando la costruzione per gli automi finiti modificata in modo che la macchina prodotto M esegua sulla pila le stesse operazioni della macchina componente N

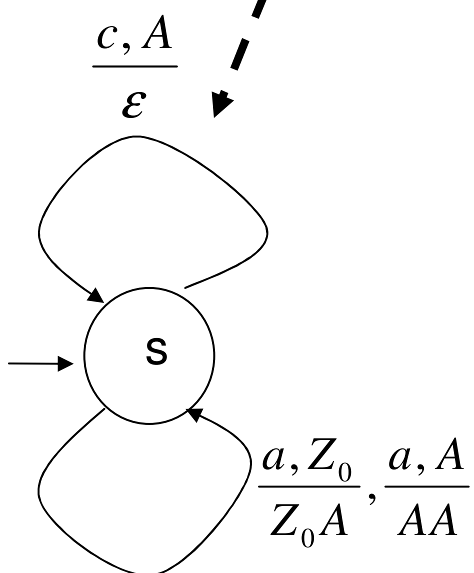
La macchina ottenuta:

- ha come stati interni il prodotto degli stati interni delle macchine componenti
- riconosce mediante **stato finale e pila vuota**
- sono finali gli stati che contengono uno stato finale dell'automa finito A
- è deterministica se sono deterministiche entrambe le macchine N e A
- riconosce a stato finale soltanto le stringhe appartenenti all'intersezione dei due linguaggi

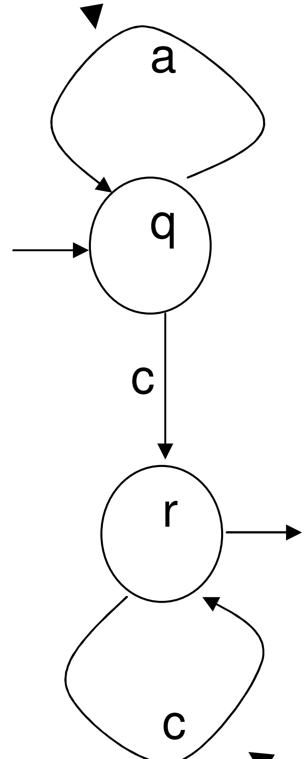
ESEMPIO

$$\boxed{L_{Dyck} \cap (a^* c^+) = \{a^n c^n \mid n \geq 1\}}$$

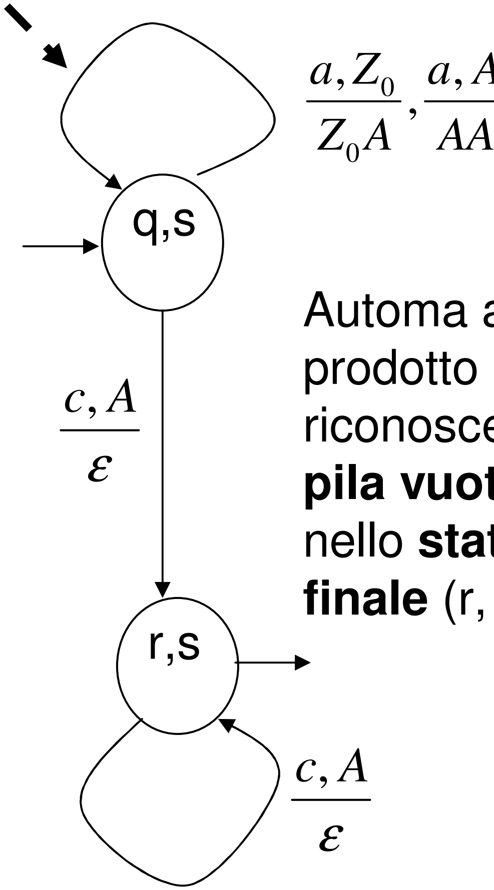
Linguaggio di Dyck con un solo nido



Automa a pila riconosce **a pila vuota**



Automa a stati finiti riconosce nello **stato finale**



Automa a pila prodotto riconosce a **pila vuota** nello **stato finale** (r, s)

AUTOMI A PILA E LINGUAGGI DETERMINISTICI (DET)

Approfondiamo lo studio dei riconoscitori deterministici e dei loro linguaggi: i più usati nei compilatori per la loro efficienza

Indeterminismo assente se funzione δ è a un solo valore e inoltre

se $\delta(q, a, A)$ è definito allora $\delta(q, \varepsilon, A)$ non è definito
se $\delta(q, \varepsilon, A)$ è definito allora $\delta(q, a, A)$ non è definito per alcun $a \in \Sigma$

Se nella funzione di transizione non è presente alcuna forma di indeterminismo L'AUTOMA è DETERMINISTICO e il LINGUAGGIO RICONOSCIUTO è detto DETERMINISTICO

NB: quindi un automa a pila **deterministico può** avere mosse spontanee

ESEMPIO

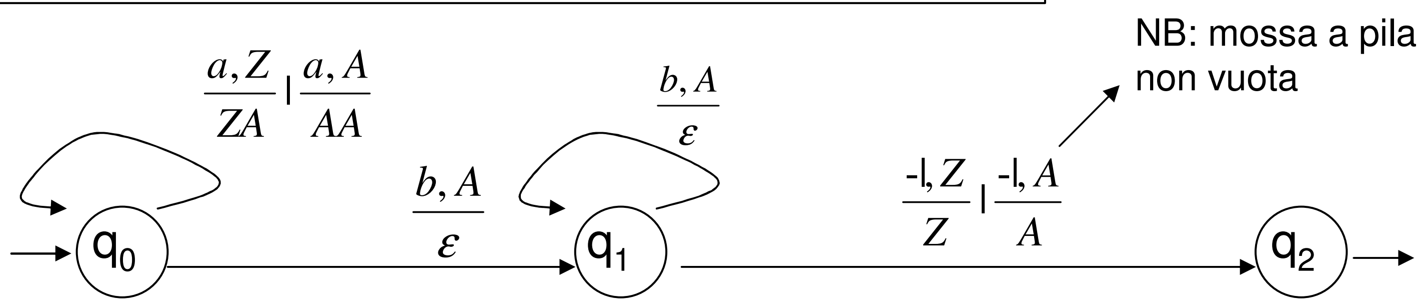
$L = \{a^n b^m \mid n \geq m > 0\}$ grammatica e automa di p.11

forme di indeterminismo:

1. $\delta(q_0, a, A) = \{(q_0, b), (q_0, bA)\}$
2. $\delta(q_0, \varepsilon, S) = \{(q_0, A)\}$ e $\delta(q_0, a, S) = \{(q_0, S)\}$

Lo stesso linguaggio è riconosciuto deterministicamente dall'automa

$$\boxed{M_2 = (\{q_0, q_1, q_2\}, \{a, b\}, \{A, Z\}, \delta, q_0, Z, \{q_2\})}$$



Intuitivamente: M impila le *a*, codificate come *A*; leggendo la prima *b* cancella una *A* e passa allo stato q_1. Poi per ogni *b* letta spila una *A*. Se vi fossero più *b* che *a* esso cadrebbe in errore. Alla lettura del terminatore, esso passa nello stato q_2, quale che sia il simbolo in cima.
Intuitivamente, non cerca (come il nondeterministico) di "indovinare" il punto in cui inizia $a^m b^m$ ma conta il numero delle *a* e verifica che il numero delle *b* non sia superiore

PROPRIETÀ DI CHIUSURA DEI LINGUAGGI DETERMINISTICI

Indicando con L, D, e R un linguaggio appartenente rispettivamente alla famiglia *LIB, DET e REG.*

Operazione	Proprietà	(Proprietà già nota)
Riflessione	$D^R \notin DET$	$D^R \in LIB$
Stella	$D^* \notin DET$	$D^* \in LIB$
Complemento	$\neg D \in DET$	$\neg L \notin LIB$
Unione	$D_1 \bigcup D_2 \notin DET, D \bigcup R \in DET$	$D_1 \bigcup D_2 \in LIB$
Concatenamento	$D_1.D_2 \notin DET, D.R \in DET$	$D_1.D_2 \in LIB$
Intersezione	$D \bigcap R \in DET$	$D_1 \bigcap D_2 \notin LIB$

NB: le operazioni tipiche dei linguaggi (R, * , ∪, ·) **NON** preservano il determinismo

<u>LINGUAGGI NON DETERMINISTICI</u>

Diversamente dal caso dei linguaggi regolari, non tutti i linguaggi liberi possono essere riconosciuti da un automa deterministico.

Considerando che:

$1)\ DET \subseteq LIB$ (l’automa deterministico è un caso particolare)

$2)\ DET \neq LIB$ (certe proprietà di chiusura valgono per una famiglia ma non per l’altra)

Si conclude che:

la famiglia DET dei linguaggi deterministici è ***strettamente*** contenuta in quella LIB dei linguaggi liberi

ESEMPIO: Unione (non deterministica) di linguaggi deterministici

$$\boxed{L = \left\{a^n b^n \mid n \geq 1\right\} \bigcup \left\{a^n b^{2n} \mid n \geq 1\right\} = L' \bigcup L''}$$

L'automa dovrebbe impilare le *a* lette e, se la stringa (ad es. *aabb*) appartiene al primo insieme, spilare una *a* alla lettura di una *b;* ma se la stringa appartiene al secondo insieme (ad es. *aabbbb*), sono due le *b* da leggere per spilare una *a*. Non potendo sapere quale sia la strada giusta, l'automa deve tentare entrambe le vie.

L',L''$\in$DET, L=L'$\cup$L'', L$\notin$DET, L$\in$LIB, quindi DET$\subseteq$LIB e DET $\neq$LIB

DETERMINISMO E INAMBIGUITÀ DEL LINGUAGGIO

Se un linguaggio è accettato da un automa deterministico, ogni frase è riconosciuta da un solo calcolo. D'altra parte la grammatica equivalente all'automa simula i calcoli dell'automa mediante le proprie derivazioni: essa genera una frase con una certa derivazione sinistra se e solo se l'automa ha un corrispondente calcolo che riconosce la stessa frase. Due diverse derivazioni sinistre corrispondono a due diversi calcoli.

PROPRIETÀ – sia M un automa a pila deterministico; allora la corrispondente grammatica di L(M), che si può ricavare meccanicamente dall'automa, non è ambigua.