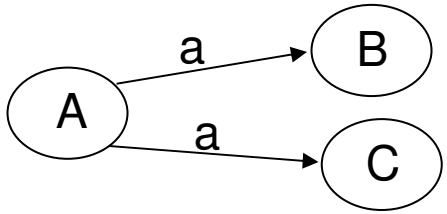
Automi non deterministici e riconoscimento

Prof. A. Morzenti

aa 2008-2009

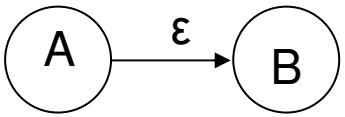
FORME DI NON DETERMINISMO (o INDETERMINISMO)

1) Piu mosse alternative per un unico ingresso

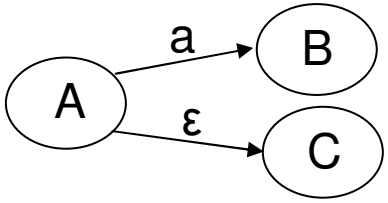


$$\delta(A,a)=\{B,C\}$$

2) mossa spontanea (o mossa ε): automa cambia stato senza "consumare" ingresso.



$$\delta(A,\varepsilon)=B$$



$$\delta(A,a)=B \quad \delta(A,\varepsilon)=B$$

3) piu` diversi stati iniziali (utile e.g. quando si fondono diversi automi…)

ANALOGIE CON GRAMMATICHE LINEARI A DESTRA

1) grammatica con due alternative $A \to aB \mid aC$ dove $a \in \Sigma$

2) grammatica con regola di copiatura $A \to B$ dove $B \in V$

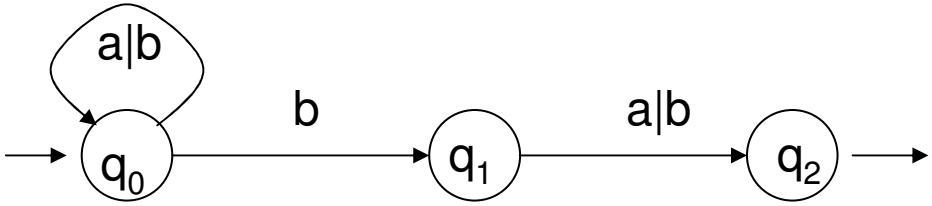
3) grammatica con piu` assiomi (utile e.g. quando si fondono diverse grammatiche …)

MOTIVAZIONI DELL'INDETERMINISMO

1) La corrispondenza tra grammatiche e automi

2) Concisione: definizioni di linguaggi più leggibili e compatte

ESEMPIO – linguaggio con penultimo carattere = b $L_2 = (a \mid b)^* b(a \mid b)$

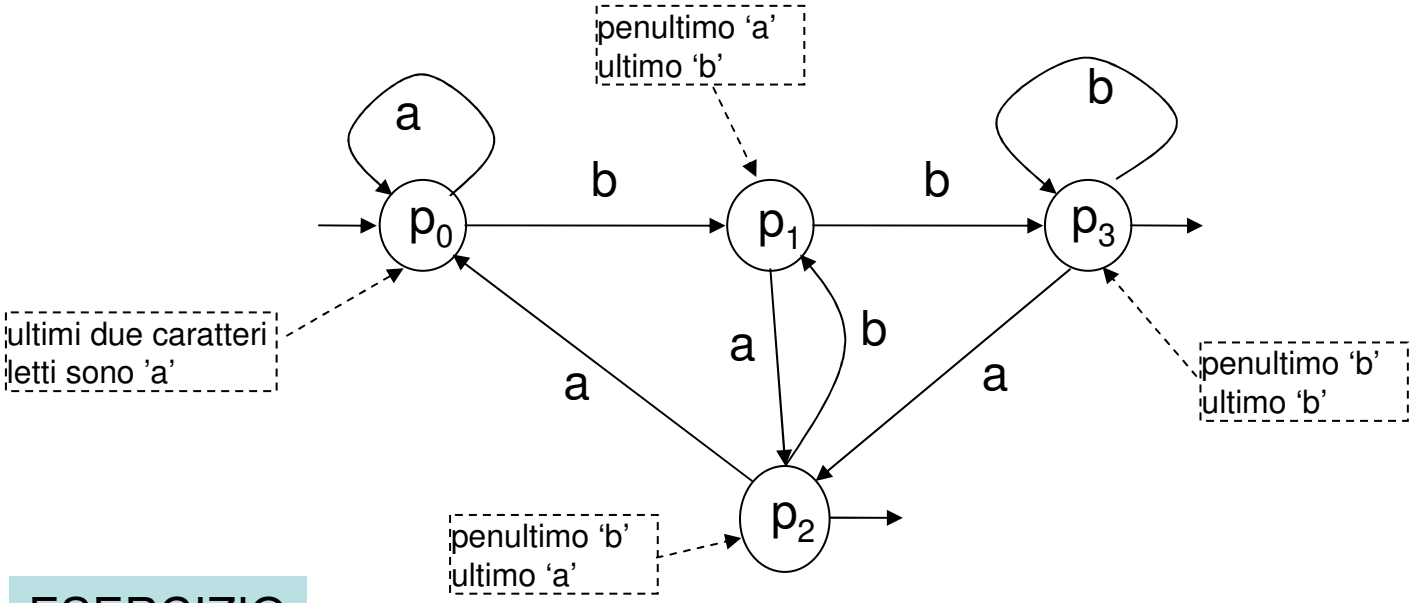


Riconoscimento di *baba. Due calcoli*
Uno accetta la stringa, l'altro no

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2$$

$$q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_0$$

Lo stesso linguaggio è accettato dall'automa deterministico M2 che però non rende altrettanto evidente la condizione che il penultimo carattere sia *b*.



ESERCIZIO

Generalizzando l'esempio, dal linguaggio L_2 al linguaggio L_k tale che il *k*-ultimo elemento, ($k \geq 2$) sia *b,* si vede che l'automa non deterministico ha $k + 1$ stati, mentre si può dimostrare che il numero di stati dell'automa deterministico minimo è dato da una funzione che cresce ***esponenzialmente*** con *k*.

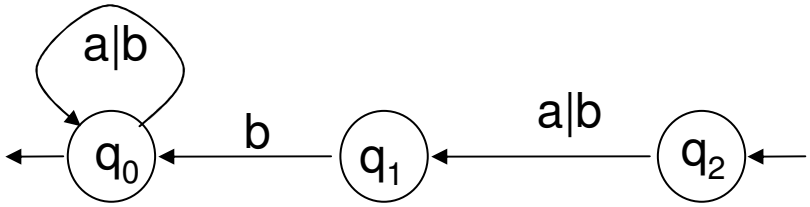
L'indeterminismo può rendere molto più concise certe definizioni.

3) Dualità sinistra – destra: Passando dal deterministico di un linguaggio L al riconoscitore del linguaggio speculare L^R (che scandisce il testo dalla fine all'inizio) e` necessario fare due operazioni
- scambio degli stati iniziale e finali
- inversione delle frecce

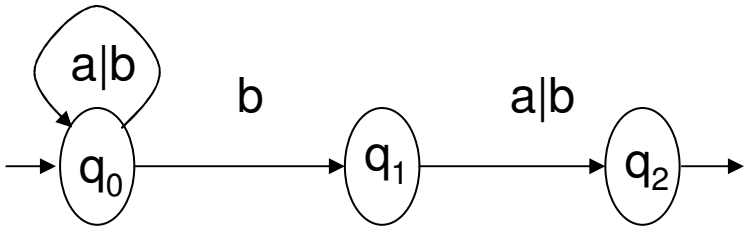
che entrambe possono far nascere indeterminismo

ESEMPIO - Il linguaggio delle stringhe aventi b come penultimo carattere è l'immagine riflessa del linguaggio delle stringhe eventi b come secondo carattere.

$$L' = \{x \mid b \text{ è il secondo carattere di } x\} \quad L_2 = (L')^R$$



'b' secondo carattere: deterministico



'b' penultimo carattere: non deterministico

4) Il passaggio attraverso automi non deterministici è conveniente nella costruzione del riconoscitore del linguaggio definito da un'espressione regolare.

RICONOSCIMENTO NON DETERMINISTICO

Un *automa non deterministico N* a stati finiti (con mosse spontanee) è definito da:

1. l'insieme degli stati Q
2. l'alfabeto terminale Σ
3. due sottoinsiemi di Q: l'***insieme*** I degli stati iniziali e l'insieme F degli stati finali
4. funzione di transizione δ a piu` valori δ: $(Q \times (\Sigma \cup \{\varepsilon\})) \rightarrow 2^Q$)

Introduciamo nozione di CALCOLO di origine q_0, di termine q_n, di lunghezza *n*, di etichetta $a_1a_2...a_n$

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} q_n \quad \textit{scritta anche come} \quad q_0 \xrightarrow{a_1 a_2 \ldots a_n} q_n$$

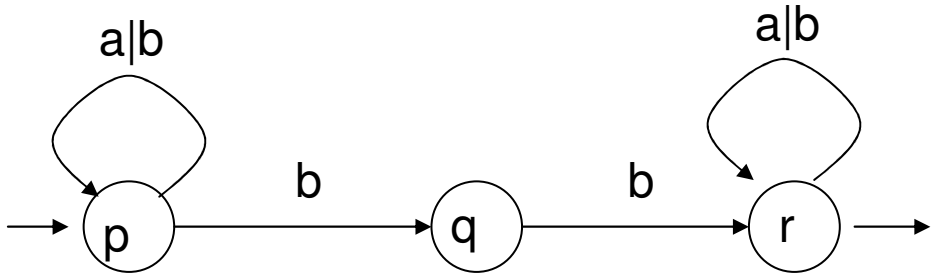
se e solo se, $\forall i$, $0 \le i < n$, $\delta(q_i, a_{i+1}) = q_{i+1}$, $a_i \in \Sigma \cup \{\varepsilon\}$

stringa x accettata (riconosciuta) dall'automa sse è etichetta di (almeno) ***un*** calcolo (NB: per una stessa stringa possibili piu` calcoli) da uno stato iniziale a uno stato finale

$$q_0 \xrightarrow{a_1 a_2 \ldots a_n} q_n,\ q_n \in F$$

Linguaggio riconosciuto dall'automa N: $L(N) = \left\{ x \mid q \xrightarrow{x} r \text{ con } q \in I, r \in F \right\}$

ESEMPIO – Ricerca di una parola di un testo
Data una parola *y*, per riconoscere se un testo la contiene lo sottoponiamo all'automa che accetta il linguaggio *(a | b)* y (a | b)*.* Consideriamo *y = bb*



La stringa *abbb* è l'etichetta di più calcoli originantisi nello stato iniziale.

$$p \xrightarrow{a} p \xrightarrow{b} p \xrightarrow{b} p \xrightarrow{b} p \qquad p \xrightarrow{a} p \xrightarrow{b} p \xrightarrow{b} p \xrightarrow{b} q$$

$$p \xrightarrow{a} p \xrightarrow{b} p \xrightarrow{b} q \xrightarrow{b} r \qquad p \xrightarrow{a} p \xrightarrow{b} q \xrightarrow{b} r \xrightarrow{b} r$$

I primi due calcoli non trovano la parola cercata. Gli ultimi due la trovano rispettivamente nelle posizioni:

$$a\underbrace{bb}b \quad \text{e} \quad ab\underbrace{bb}$$

FUNZIONE DI TRANSIZIONE (PER STRINGHE)

Per un automa indeterministico N = (Q, Σ, δ, I, F) si ha δ: (Q × (Σ∪{ε})*) → 2^Q)

$$\text{per un carattere}: \quad \delta(q,a)=\{p_1, p_2, ...p_k\}$$

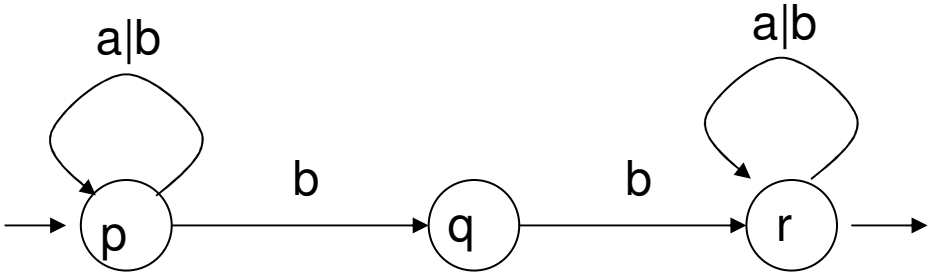
$$\text{per la stringa vuota}: \quad \forall q \in Q \ \delta(q,\varepsilon)=\{q\}$$

$$\text{per una stringa qualsiasi}: \quad \forall q \in Q, \forall y \in \Sigma^* \quad \delta(q,y)=\{p \mid q \xrightarrow{y} p\}$$

Definizione di linguaggio accettato dall'automa N usando la funzione δ (e considerando anche le ε-mosse).

$$L(N)=\left\{x \in \Sigma^* \mid \exists q \in I : \delta(q,x) \bigcap F \neq \varnothing\right\}$$

ESEMPIO (Ricerca di una parola in un testo – continua)



$$\delta(p,a) = [p],$$
$$\delta(p,ab) = [p,q],$$
$$\delta(p,abb) = [p,q,r]$$

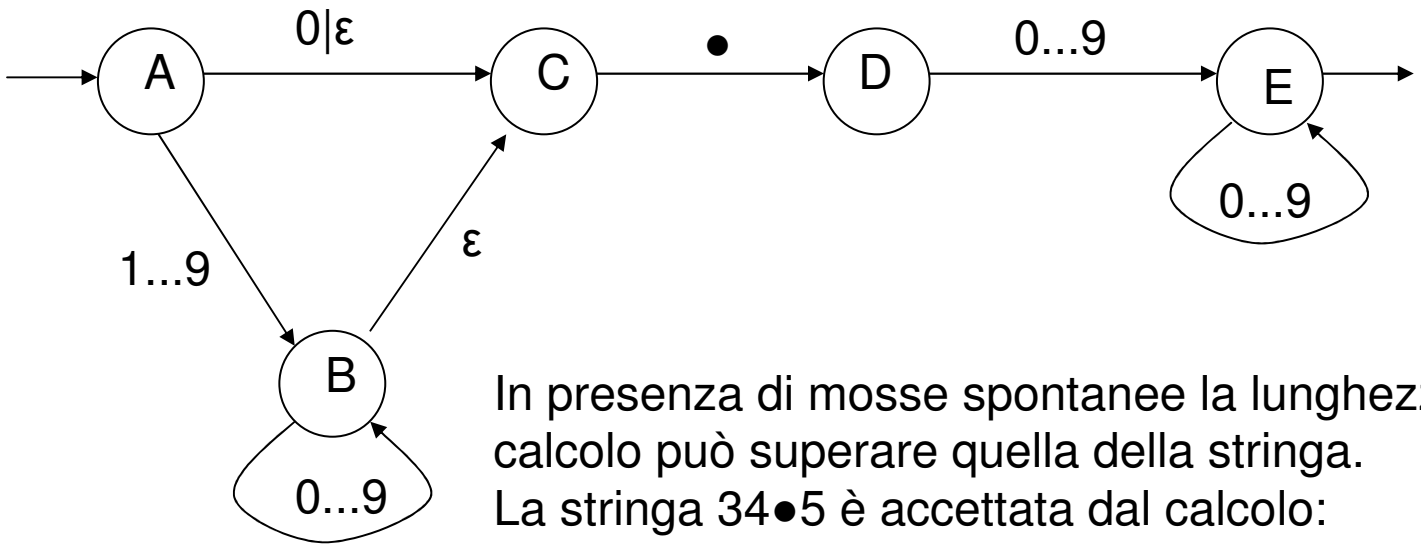
AUTOMI CON MOSSE SPONTANEE

Le mosse spontanee sono rappresentabili in un diagramma stato-transizioni con un arco etichettato ε (ε-arco)

Con l'uso di ε-archi è facile costruire i riconoscitori di linguaggi ottenuti per composizione regolare di altri linguaggi.

ESEMPIO – costanti decimali (con o senza 0 prima del punto, senza zeri prima di altre cifre nella parte intera)

$$L = \left(0 \mid \varepsilon \mid \left((1..9)(0..9)^{*}\right)\right)\bullet(0..9)^{+}$$



In presenza di mosse spontanee la lunghezza del calcolo può superare quella della stringa.
La stringa 34●5 è accettata dal calcolo:

$$A \xrightarrow{3} B \xrightarrow{4} B \xrightarrow{\varepsilon} C \xrightarrow{\square} D \xrightarrow{5} E$$

UNICITÀ DELLO STATO INIZIALE. Nelle definizioni l'automa indeterministico può avere più stati iniziali, ma è facile ottenere un automa equivalente con stato iniziale unico. Si aggiunge uno stato iniziale q_0 e le ε-mosse che da esso portano agli (ex) stati iniziali dell'automa da trasformare.

Un calcolo del nuovo automa accetta una stringa se, e soltanto se, anche il vecchio automa la accetta. Si potranno poi eliminare le mosse aggiunte, nel modo che si vedrà.

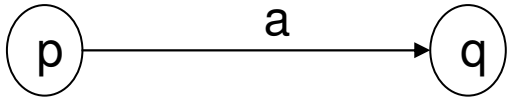
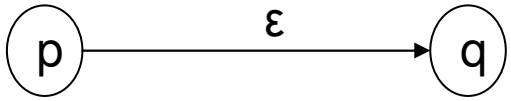
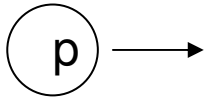
CORRISPONDENZA TRA AUTOMA E GRAMMATICA

Gia` vista trasformazione da automa a grammatica
Ora vediamo trasformazione da grammatica ad automa

Si supponga:

G = (V, Σ, P, S) lineare a dx con regole strettamente unilineari (1 solo term./regola)

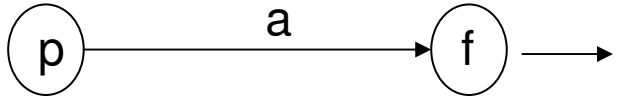
N = (Q, Σ, δ, q_0, F) è l'automa, che si può supporre avere stato iniziale unico.

Grammatica lineare a destra	*Automa finito*
1. Alfabeto non terminale V	Insieme degli stati Q = V
2. Assioma S	Stato iniziale q_0 = S
3. $p \to aq,\ a \in \Sigma \text{ e } p,q \in V$	
4. $p \to q \text{ dove } p,q \in V$	
5. $p \to \varepsilon$	Stato finale 

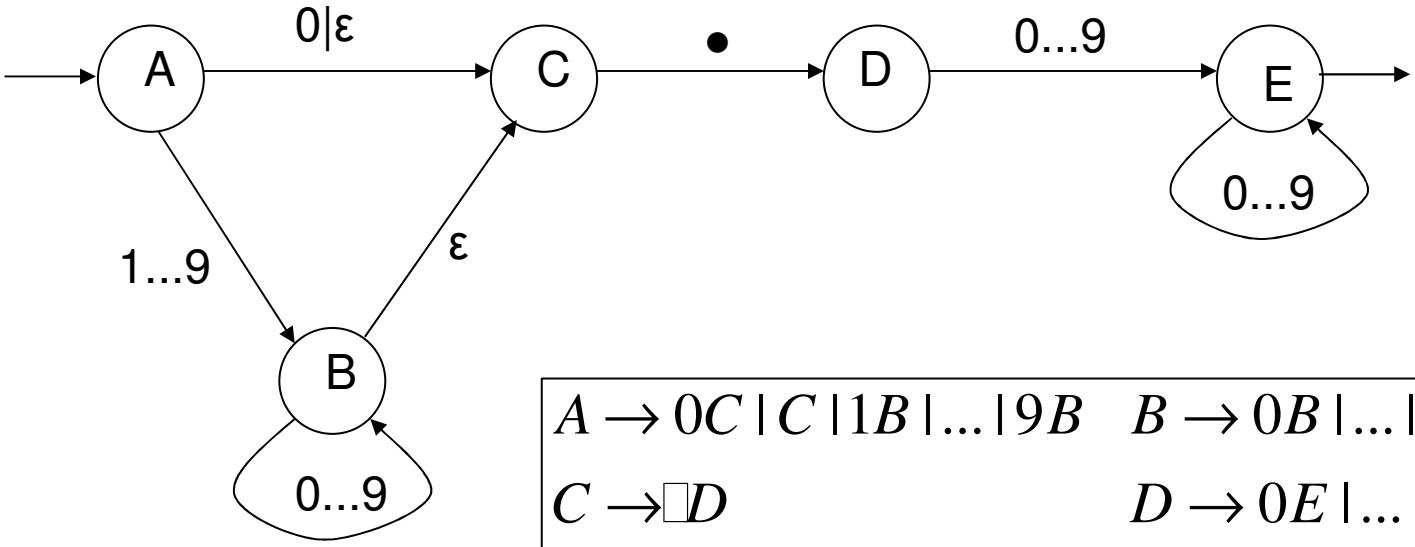
OGNI DERIVAZIONE DELLA GRAMMATICA corrisponde a un CALCOLO DELL'AUTOMA, e viceversa; di conseguenza i due modelli definiscono lo stesso linguaggio.

UN LINGUAGGIO È RICONOSCIUTO DA UN AUTOMA FINITO SE E SOLO SE È GENERATO DA UNA GRAMMATICA UNILINEARE.

Se la grammatica possiede anche delle regole terminali del tipo $p \rightarrow a$ con $a \in \Sigma$, l'automa conterrà anche uno stato finale *f*, distinto da quelli corrispondenti ai simboli non terminali della grammatica, e la mossa:

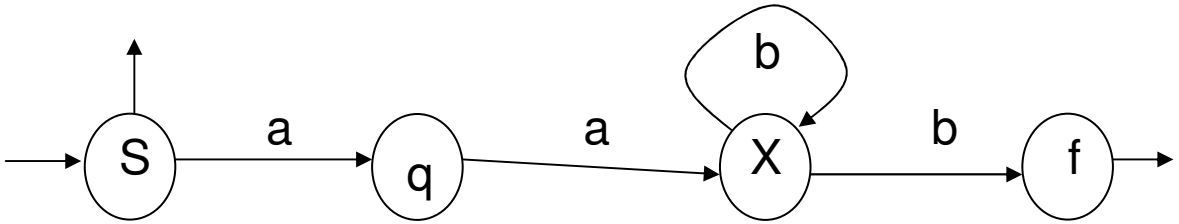


ESEMPIO – Equivalenza tra grammatiche lineari a destra e automi



$$A \rightarrow 0C \mid C \mid 1B \mid ... \mid 9B \qquad B \rightarrow 0B \mid ... \mid 9B \mid C$$
$$C \rightarrow \square D \qquad D \rightarrow 0E \mid ... \mid 9E$$
$$E \rightarrow 0E \mid ... \mid 9E \mid \varepsilon \qquad \text{dove } A \text{ è l'assioma}$$

Come trattare grammatica lineare a destra ma non strettamente lineare?
Si vede facilmente su un esempio:



$$S \rightarrow aaX \mid \varepsilon$$
$$X \rightarrow bX \mid b$$

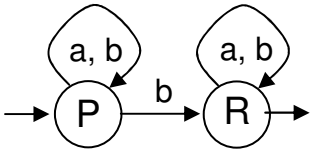
DEFINIZIONE: un automa e` ambiguo sse accetta una frase con piu` calcoli diversi

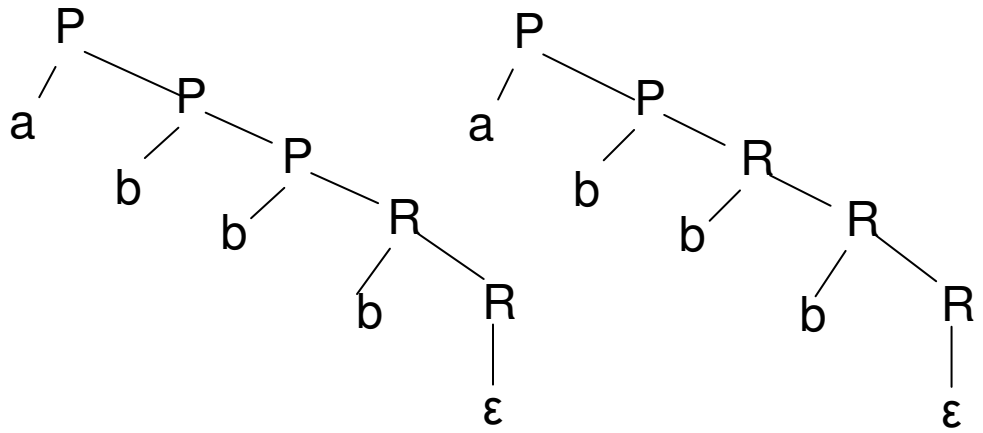
Dalla corrispondenza biunivoca tra calcoli (automa) e derivazioni (grammatica) segue che un automa è ambiguo se, e soltanto se, la grammatica lineare a destra corrispondente è ambigua (genera una frase con due alberi diversi).

ESEMPIO – Riconoscimento di una parola in un testo – stringa *abbb*

P → aP | bP | bR

R → aR | bR | ε





GRAMMATICA LINEARE A SINISTRA E AUTOMA

$$A \to Ba \quad A \to B \quad A \to \varepsilon$$

$$L^R = (L(G))^R \quad \text{generato da } G_R$$

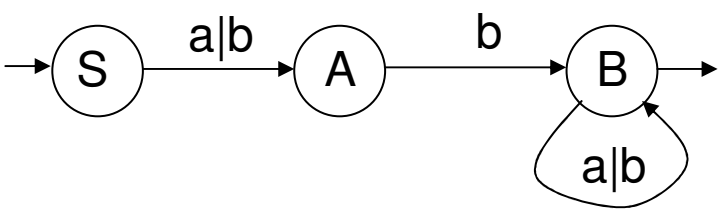
ESEMPIO – Linguaggio in cui il penultimo carattere è *b*.

$$G: \; S \to Aa \mid Ab \quad A \to Bb \quad B \to Ba \mid Bb \mid \varepsilon$$

$$G_R: \; S \to aA \mid bA \quad A \to bB \quad B \to aB \mid bB \mid \varepsilon$$

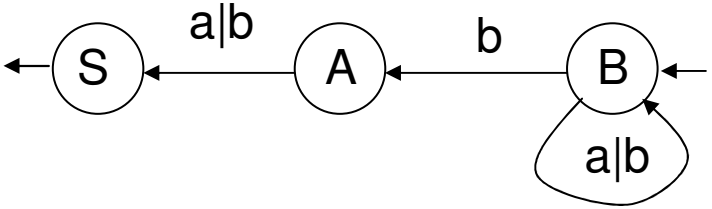
produzioni ribaltate

riconoscitore di $(L(G))^R$:



il secondo carattere è *b*

riconoscitore di L(G):



il penultimo carattere è *b*

DALL'AUTOMA ALL'ESPRESSIONE REGOLARE DIRETTAMENTE
IL METODO BMC (Brzozowski e McCluskey)

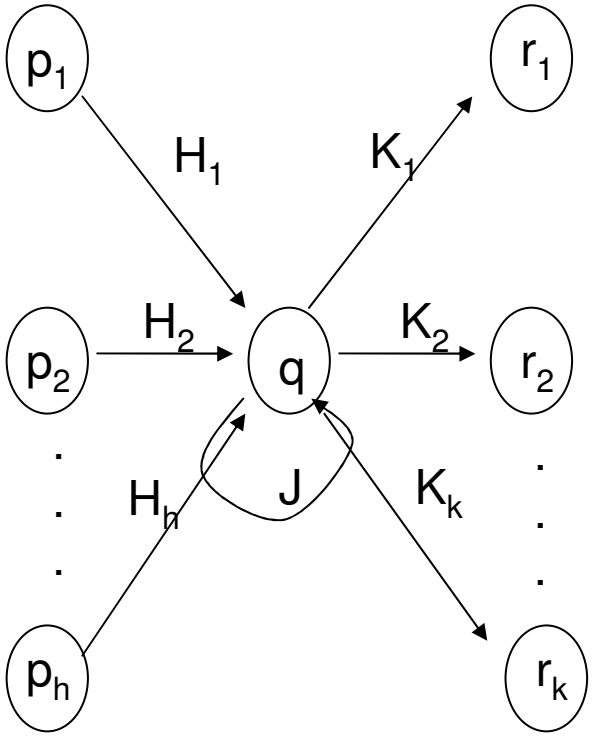
Si assume che lo stati iniziale *i* sia unico e privo di archi entranti, e lo stato finale *t* sia unico e privo di archi uscenti.

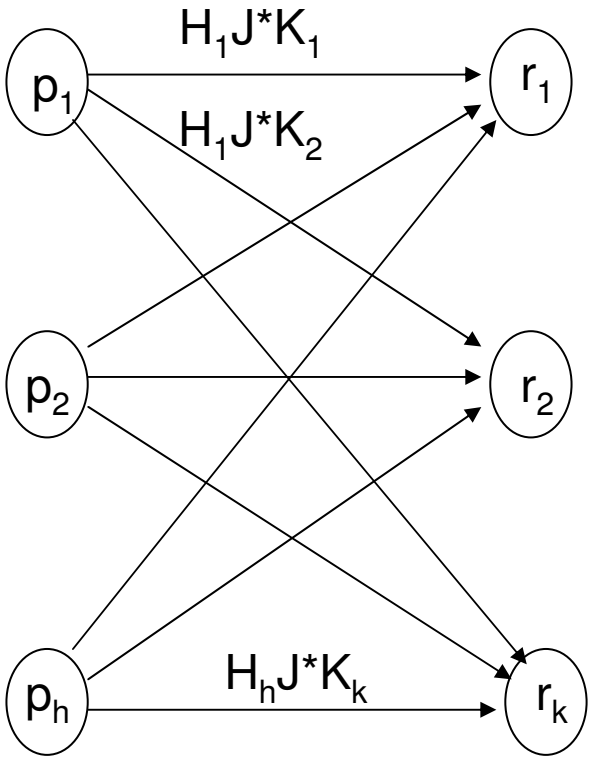
Se così non e`si può creare un nuovo stato iniziale e un nuovo stato finale e collegarli con mosse spontanee.

STATI INTERNI sono gli stati diversi da *i* e da *t*.

Si costruisce l'AUTOMA GENERALIZZATO: un automa equivalente a quello dato, dove gli archi possono essere etichettati anche con espressioni regolari.

Si eliminano uno alla volta gli stati ***interni***, aggiungendo mosse compensatorie che preservano il linguaggio riconosciuto, etichettate con e.r.; fino a quando restano solo *i* e *t*. A quel punto l'etichetta dell'arco che va da *i* a *t* è la e.r. del linguaggio.



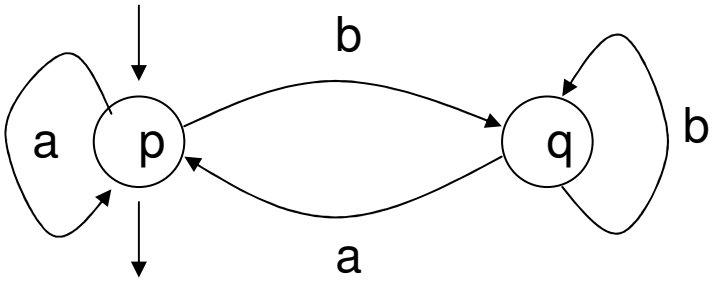


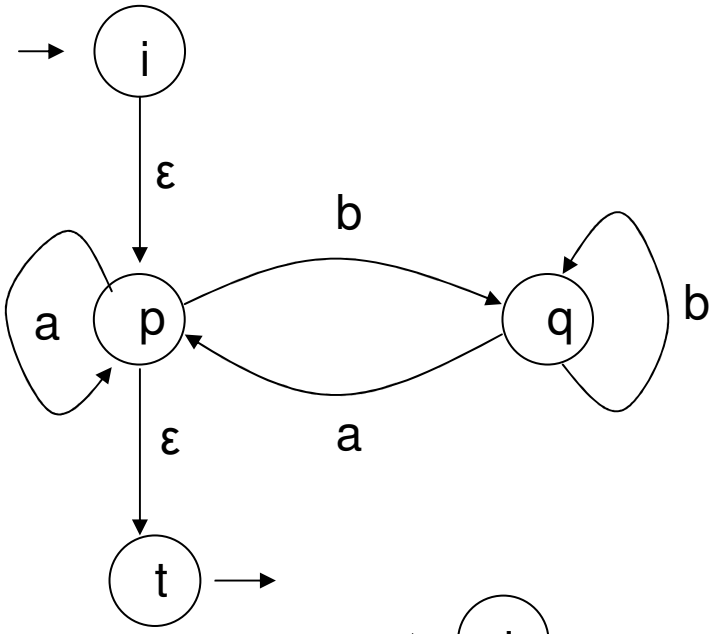
Per ogni coppia di stati p_i, r_j, vi è l'arco: $p_i \xrightarrow{H_i J^* K_j} r_j$

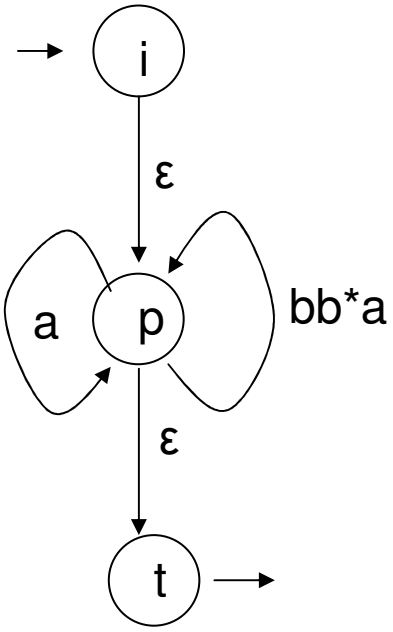
NB: Alcuni stati p_i, r_j potrebbero essere Coincidenti.

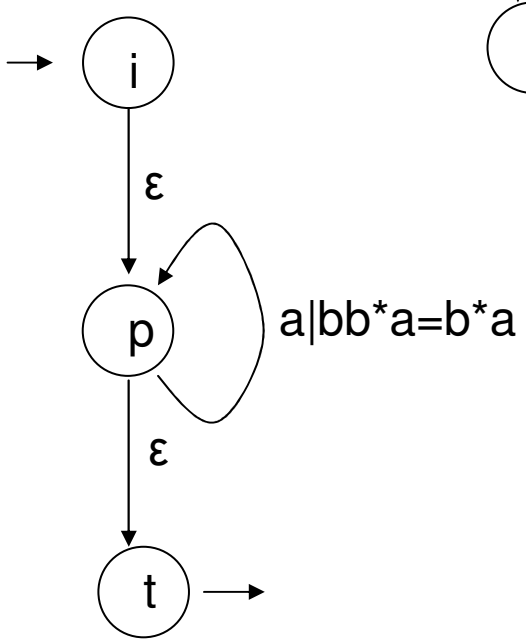
L'ordine di eliminazione degli stati interni è irrilevante. Ordini diversi possono produrre e.r. equivalenti ma di diversa complessità.

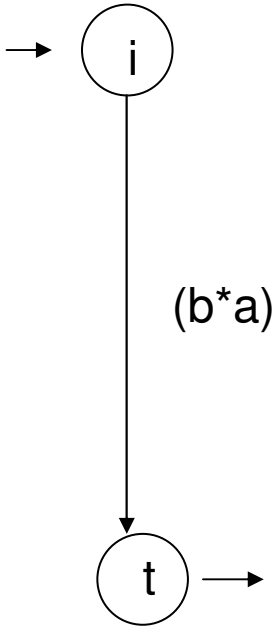
ESEMPIO- normalizzazione e applicazione di BMC nell'ordine q, p.











ELIMINAZIONE DELL'INDETERMINISMO – PROCEDIMENTO COSTRUTTIVO

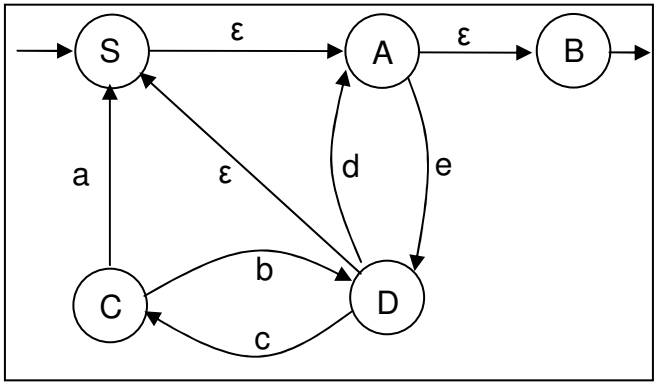
La versione finale del riconoscitore di un linguaggio deve quasi sempre essere deterministica, per ragioni di efficienza.

Ogni automa indeterministico può essere trasformato in uno deterministico equivalente e (corollario) ogni grammatica unilineare ammette una grammatica equivalente non ambigua.

LA TRASFORMAZIONE di un automa indeterministico a uno deterministico procede in DUE FASI:

1. ELIMINAZIONE DELLE MOSSE SPONTANEE: Si passa per la grammatica unilineare destra equivalente all'automa: le ε-mosse corrispondono a regole di copiatura: si applica la trasformazione grammaticale che le elimina (già vista).
2. SOSTITUZIONE DI PIÙ TRANSIZIONI NON DETERMINISTICHE con una sola (costruzione delle parti finite – i nuovi stati introdotti corrispondono a sottoinsiemi dell'insieme degli stati).

ESEMPIO della prima fase: eliminazione delle ε-mosse



automa ⇒ grammatica

$$S \to A \qquad A \to B \mid eD \qquad B \to \varepsilon$$
$$C \to aS \mid bD \qquad D \to S \mid cC \mid dA$$

$$Copia(X) = \{Y \in V \mid X \overset{*}{\Rightarrow} Y\}$$

	$Copia$
S	S, A, B
A	A, B
B	B
C	C
D	D, S, A, B

elimina regole di copiatura
S→A, A→B, D→S

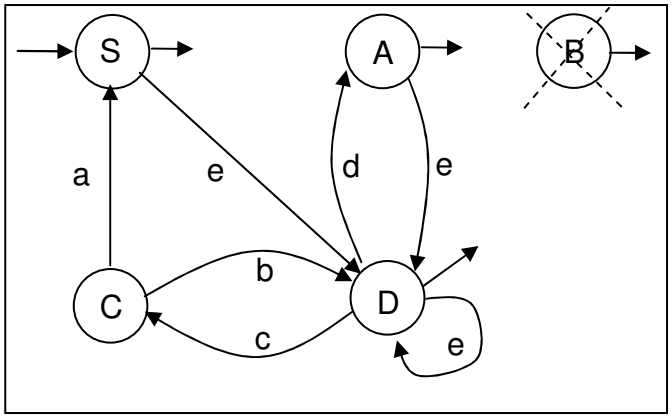
$$A \to eD \qquad B \to \varepsilon$$
$$C \to aS \mid bD \qquad D \to cC \mid dA$$

aggiungi nuove regole

per S: $S \to eD$ (per via di A) $S \to \varepsilon$ (per via di B)
per A: $A \to \varepsilon$ (per via di B)
per D: $D \to eD$ (per via di A) $D \to \varepsilon$ (per via di B)

$$S \to \varepsilon \mid eD \qquad A \to \varepsilon \mid eD \qquad B \to \varepsilon$$
$$C \to aS \mid bD \qquad D \to \varepsilon \mid eD \mid cC \mid dA$$

grammatica ⇒ automa



DETERMINIZZAZIONE CON L'INSIEME DELLE PARTI FINITE

Dato l'automa NDA non deterministico privo di mosse spontanee, si costruisce l'automa deterministico equivalente DA.

IDEA: DA "simula" NDA; ogni stato di DA "contiene" un ***insieme*** di stati raggiungibili da un calcolo di NDA che parta da uno stato iniziale.

Se in NDA vi sono le mosse: $q_0 \overset{a}{\to} p_1,\ q_0 \overset{a}{\to} p_2,\ \ldots\ q_0 \overset{a}{\to} p_k$

si costruisce in DA un nuovo stato collettivo $\{p_1, p_2, \ldots\ p_k\}$
per indicare l'incertezza tra i *k* stati

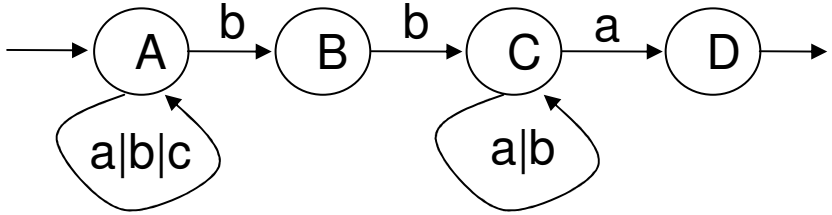
si costruiscono poi le transizioni uscenti dagli stati collettivi:

se $p_1 \overset{a}{\to} \{q_1, q_2, \ldots\ q_{k_1}\},\ \ p_2 \overset{a}{\to} \{r_1, r_2, \ldots\ r_{k_2}\},\ \ \ldots$

allora $\{p_1, \mathrm{p}_2, \ldots\} \overset{a}{\to} \{q_1, q_2, \ldots\ q_{k_1}\} \cup \{r_1, r_2, \ldots\ r_{k_2}\} \cup \ldots = \{q_1, q_2, \ldots\ q_{k_1}, r_1, r_2, \ldots\ r_{k_2}, \ldots\}$

Se in DA non esiste lo stato collettivo di arrivo della transizione, lo si crea.

ESEMPIO: NDA sia



$\delta(A,b)=\{A,B\}$ creiamo stato *{A,B}*

$\delta(\{A,B\},a)=\delta(A,a)\cup\delta(B,a)=\{A\}\cup\{\}=\{A\}$

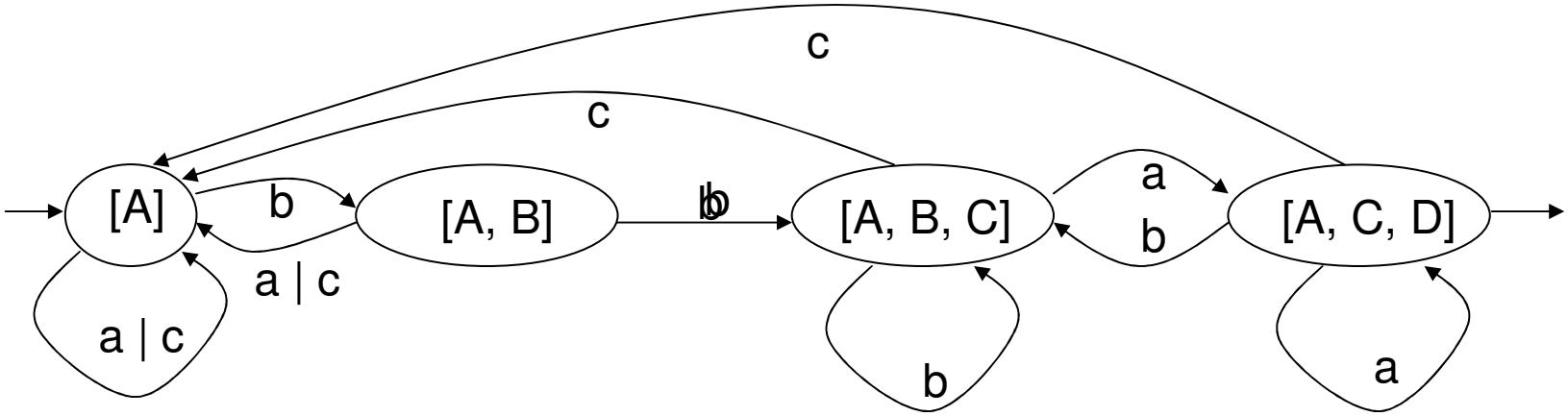
$\delta(\{A,B\},b)=\delta(A,b)\cup\delta(B,b)=\{A,B\}\cup\{C\}=\{A,B,C\}$ creiamo stato *{A,B,C}*

$\delta(\{A,B,C\},a)=\delta(A,a)\cup\delta(B,a)\cup\delta(C,a)=\{A\}\cup\{\}\cup\{C,D\}=\{A,C,D\}$ creiamo stato *{A,C,D}*

$\delta(\{A,C,D\},a)=\delta(A,a)\cup\delta(C,a)\cup\delta(D,a)=\{A\}\cup\{C,D\}\cup\{\}=\{A,C,D\}$ etc.....

Nota: non tutti i sottoinsiemi di Q sono raggiungibili (ad esempio [A, C])

Costruzione termina quando, considerati tutti gli ingressi per tutti gli stati, non viene trovato alcuno stato nuovo. Sono finali gli stati di DA che contengono stati finali di NDA



ALGORITMO DELL'INSIEME DELLE PARTI FINITE

L'automa deterministico DA=<Q', Σ, δ', q_o,F'> equivalente a NDA= (Q, Σ, δ, I, F) ha:

1. gli stati Q' = P(Q) = 2^Q, l'insieme delle parti di Q
2. gli stati finali F', quelli contenenti uno stato finale di N: $F' = \{p' \in Q' \mid p' \cap F \neq \varnothing\}$
3. lo stato iniziale q_0 = I (l'insieme degli stati iniziali di NDA)
4. funzione di transizione δ':

$$\forall p' \in Q' \ \forall a \in \Sigma \ \delta'(p', a) = \bigcup_{q \in p'} \delta(q, a)$$

$$\text{o, detto altrimenti, } p' \xrightarrow{a} \{s \mid q \in p' \wedge (q \xrightarrow{a} s \text{ è una transizione di NDA})\}$$

NOTE:

1. se q va in q_{err} tramite a, lo stato di errore non va aggiunto allo stato collettivo, i calcoli che portano allo stato di errore non riconoscono alcuna stringa e si possono ignorare
2. gli stati di Q' sono i sottoinsiemi di Q e la cardinalità di Q' è, nel caso peggiore, esponenziale rispetto a |Q| (si ha quindi in generale un maggiore dimensione dell'automa deterministico)
3. DA spesso ha degli stati irraggiungibili dallo stato iniziale, dunque inutili; si disegnano solo gli stati raggiungibili partendo dallo stato iniziale (mediante costruzione incrementale esemplificata prima)

IL PROCEDIMENTO È VALIDO INFATTI una stringa x è accettata da DA se, e solo se, è accettata da NDA.

A) Se un calcolo di NDA accetta x, esiste un cammino etichettato x dallo stato iniziale q_0 a uno stato finale q_f. L'algoritmo garantisce che anche in DA esista un (solo) percorso etichettato x da $[q_0]$ a uno stato [....,q_f,....] contenente q_f.

B) Se x è l'etichetta di un calcolo valido di DA, da q_0 a uno stato finale $p \in F'$, allora per costruzione p contiene almeno uno stato finale q_f di NDA. Per costruzione esiste allora un cammino di etichetta x da q_0 a q_f.

PROPRIETÀ – ogni linguaggio a stati finiti è riconosciuto da un automa deterministico. L'algoritmo di riconoscimento opera quindi in tempo reale.

COROLLARIO: per ogni linguaggio riconosciuto da un automa a stati finiti, esiste una grammatica unilineare priva di ambiguità, quella che corrisponde in modo naturale all'automa deterministico. Per i linguaggi regolari si può eliminare l'ambiguità, costruendo il riconoscitore deterministico e la grammatica a esso equivalente.