

Modelli per i linguaggi

Modelli adeguati per riconoscere/accettare, tradurre, calcolare linguaggi

- “Ricevono” una stringa di ingresso e la elaborano

- Modelli operazionali (Automati)

Modelli adeguati per descrivere come generare un linguaggio

- Insiemi di regole per costruire le frasi di un linguaggio

- Modelli generativi (Grammatiche)

Grammatiche (1)

- I modelli generativi producono stringhe
 - grammatica (o sintassi)
- Una grammatica è un insieme di regole per costruire le frasi di un linguaggio
 - Si applica a qualunque nozione di linguaggio
- Una grammatica formale genera stringhe di un linguaggio attraverso un processo di riscrittura

Riscrittura

- La riscrittura è importante in molti campi
 - Matematica
 - Informatica
 - Logica
- Consiste di un insieme di metodi per sostituire sotto-parti di una formula con altre parti
 - Potenzialmente non deterministica

Esempi

- Formule semanticamente equivalenti nella logica proposizionale
 - $A \wedge B$ può essere sostituito da $\neg(\neg A \vee \neg B)$
 - $\neg A \vee B$ può essere sostituito da $A \Rightarrow B$
 - ...
- Esempi di tautologie in FOL
 - Possiamo riscrivere la tautologia $\neg A \vee A$ sostituendo A con una formula ben formata della logica proposizionale o di FOL

Regole linguistiche (1)

- I linguaggi naturali sono spiegati tramite regole quali:
 - Un periodo è composto da una o più proposizioni collegate da congiunzioni o punteggiatura
 - Una proposizione è composta da un soggetto, un predicato ed eventuali complementi
 - Il soggetto può essere un nome, un pronome, o ...
- I linguaggi di programmazione sono espressi in modo simile:
 - Un programma consiste di una parte dichiarativa e una esecutiva
 - La parte esecutiva consiste di una sequenza di istruzioni
 - Un'istruzione può essere ...

Regole linguistiche (2)

- In generale, una regola linguistica descrive un “oggetto principale”
 - Esempi: un libro, un programma, un messaggio, ...come una sequenza di “componenti”
- Ogni “componente” è “raffinato” sostituendolo con componenti più dettagliati, e così via... fino ad ottenere una sequenza di elementi di base

Grammatiche (2)

- Una grammatica è una regola linguistica
- E' composta da
 - oggetto principale: il simbolo iniziale
 - componenti: i simboli non terminali
 - elementi di base: simboli terminali
 - regole di raffinamento: produzioni
- Formalmente?

Citazioni

“A grammar can be regarded as a device that enumerates the sentences of a language”

“A grammar of L can be regarded as a function whose range is exactly L ”

- Noam Chomsky, *On Certain Formal Properties of Grammars*, Information and Control, Vol 2, 1959

Definizione

- Una grammatica è una tupla $\langle V_N, V_T, P, S \rangle$ in cui
 - V_N è l'alfabeto non terminale
 - V_T è l'alfabeto terminale
 - $V = V_N \cup V_T$
 - $S \in V_N$ è un particolare elemento di V_N chiamato assioma o simbolo iniziale
 - $P \subseteq V_N^+ \times V^*$ è l'insieme (finito) di regole di riscrittura o produzioni
- Una grammatica $G = \langle V_N, V_T, P, S \rangle$ genera un linguaggio sull'alfabeto V_T

Produzioni

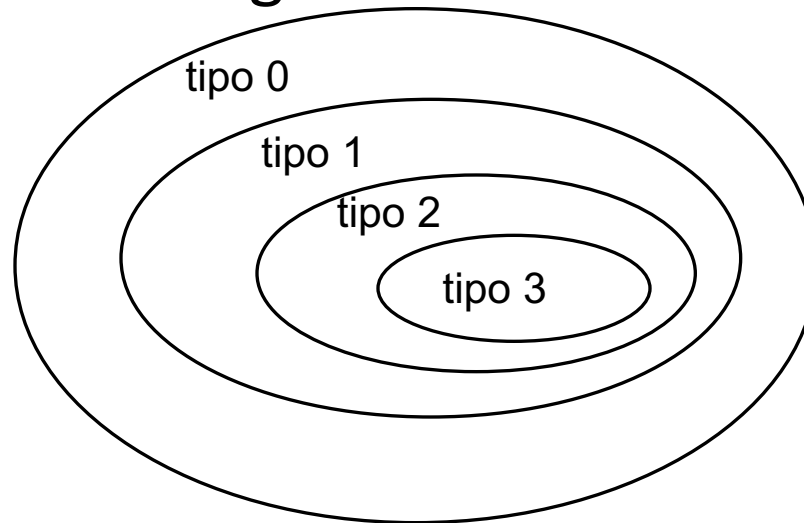
- Una produzione è un elemento di $V_N^+ \times V^*$
 - Ossia una coppia $\langle \alpha, \beta \rangle$, dove $\alpha \in V_N^+$ e $\beta \in V^*$
- Generalmente indichiamo una produzione come $\alpha \rightarrow \beta$
 - α è una sequenza di simboli non terminali
 - Altre definizioni che includono anche i terminali sono possibili
 - β è una sequenza (potenzialmente vuota) di simboli (terminali o non terminali)

Esempio

- $V_N = \{S, A, B, C, D\}$
 - $V_T = \{a, b, c\}$
 - S è il simbolo iniziale
 - Non è obbligatorio che si chiami S
 - $P = \{$
 - $S \rightarrow AB,$
 - $BA \rightarrow cCD,$
 - $CBS \rightarrow ab,$
 - $A \rightarrow \varepsilon\}$
- Il linguaggio generato è sull'alfabeto $\{a, b, c\}$

Gerarchia di Chomsky (1959)

- Le grammatiche sono classificate a seconda dalla forma delle loro produzioni
- Chomsky identificò 4 tipi di grammatiche
 - E' una classificazione di interesse storico
 - Noi useremo in seguito nomi e definizioni più agevoli



Tipi della gerarchia di Chomsky

- Le grammatiche di tipo 3 restringono le produzioni a un singolo non terminale a sinistra; a destra c'è un singolo terminale, eventualmente seguito (o preceduto, ma non entrambe le cose nella stessa grammatica) da un singolo non terminale
 - La regola $S \rightarrow \epsilon$ è consentita se S non appare a destra in nessuna regola
- Le grammatiche di tipo 2 hanno regole della forma $A \rightarrow \gamma$ dove A è un non terminale e γ è una stringa di terminali e non terminali

Tipi della gerarchia di Chomsky

- Le grammatiche di tipo 1 hanno regole della forma $\alpha A \beta \rightarrow \alpha \gamma \beta$, dove A è un non terminale e α e β sono stringhe di non terminali e γ è una stringa di terminali e non terminali.
 - γ dev'essere non vuota
 - La regola $S \rightarrow \epsilon$ è consentita se S non appare a destra in nessuna regola
 - Secondo altre definizioni, si ammette che α e β contengano anche terminali
- Le grammatiche di tipo 0 includono tutte le grammatiche formali

Relazione di derivazione immediata

$\alpha \Rightarrow \beta$ (β è ottenuta da α mediante derivazione immediata)

– $\alpha \in V^+$ e $\beta \in V^*$

se e solo se

$\alpha = \alpha_1 \alpha_2 \alpha_3$, $\beta = \alpha_1 \beta_2 \alpha_3$ e $\alpha_2 \rightarrow \beta_2 \in P$

cioè α_2 è riscritta come β_2 nel contesto $\langle \alpha_1, \alpha_3 \rangle$

Esempio

Nella grammatica G

– $V_N = \{S, A, B, C, D\}$

– $V_T = \{a, b, c\}$

– S è il simbolo iniziale

– $P = \{S \rightarrow AB, BA \rightarrow cCD, CBS \rightarrow ab, A \rightarrow \varepsilon\}$

- $aa\mathbf{BAS} \Rightarrow aac\mathbf{CDS}$
- $bc\mathbf{CBS}Add \Rightarrow bc\mathbf{ab}Add$

Linguaggio generato da una grammatica

- Data una grammatica $G = \langle V_N, V_T, P, S \rangle$

$$\forall x (x \in L(G) \Leftrightarrow x \in V_T^* \wedge S \Rightarrow^+ x)$$

- Informalmente il linguaggio generato da una grammatica G è l'insieme di tutte le stringhe
 - che consistono di soli simboli terminaliche possono essere derivate da S
 - in un qualunque numero di passi (almeno uno)

Esempio 1

- $G_1 = \langle \{S, A, B\}, \{a, b, 0\}, P, S \rangle$
 - $P = \{S \rightarrow aA, A \rightarrow aS, S \rightarrow bB, B \rightarrow bS, S \rightarrow 0\}$
- Alcune derivazioni
 - $S \Rightarrow 0$
 - $S \Rightarrow aA \Rightarrow aaS \Rightarrow aa0$
 - $S \Rightarrow bB \Rightarrow bbS \Rightarrow bb0$
 - $S \Rightarrow aA \Rightarrow aaS \Rightarrow aabB \Rightarrow aabbS \Rightarrow aabb0$
- Semplice generalizzazione: $L(G_1) = \{aa, bb\}^* \cdot \{0\}$

Esempio 2

- $G_2 = \langle \{S\}, \{a, b\}, \{S \rightarrow aSb \mid ab\}, S \rangle$
 - $\{S \rightarrow aSb \mid ab\}$ è un'abbreviazione di $\{S \rightarrow aSb, S \rightarrow ab\}$
- Alcune derivazioni
 - $S \Rightarrow ab$
 - $S \Rightarrow aSb \Rightarrow aabb$
 - $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb$
- Semplice generalizzazione: $L(G_2) = \{a^n b^n \mid n > 0\}$
 - $L(G_2) = \{a^n b^n \mid n \geq 0\}$ se sostituiamo $S \rightarrow ab$ con $S \rightarrow \epsilon$

Esempio 3

- $G_3 = \langle \{S, A, B, C, D\}, \{a, b, c\}, P, S \rangle$
 - $P = \{S \rightarrow aACD, A \rightarrow aAC \mid \varepsilon, B \rightarrow b, CD \rightarrow BDc, CB \rightarrow BC, D \rightarrow \varepsilon\}$
- Alcune derivazioni
 - $S \Rightarrow a\mathbf{ACD} \Rightarrow a\mathbf{CD} \Rightarrow aBDc \Rightarrow^* abc$
 - $S \Rightarrow a\mathbf{ACD} \Rightarrow aa\mathbf{ACCD} \Rightarrow aa\mathbf{CCD} \Rightarrow aa\mathbf{CBDc} \Rightarrow aa\mathbf{BCDc}$
 $\Rightarrow aab\mathbf{CDc} \Rightarrow aab\mathbf{BDcc} \Rightarrow aabb\mathbf{Dcc} \Rightarrow aabbcc$
 - $S \Rightarrow a\mathbf{ACD} \Rightarrow aa\mathbf{ACCD} \Rightarrow aa\mathbf{CCD} \Rightarrow aa\mathbf{CC}$

Alcune domande spontanee

- Qual è l'uso pratico delle grammatiche?
- Quali linguaggi si possono ottenere tramite le grammatiche?
- Che relazione c'è tra automi e grammatiche?
 - E tra i linguaggi generati dalle grammatiche e quelli accettati dagli automi?
 - E la gerarchia di Chomsky?

Alcune risposte

- Le grammatiche della gerarchia di Chomsky hanno nomi alternativi
 - Grammatiche di tipo 3: regolari
 - Grammatiche di tipo 2: non contestuali
 - Grammatiche di tipo 1: dipendenti dal contesto
 - Grammatiche di tipo 0: generali (o non ristrette)
- Correlazioni
 - Grammatiche regolari – linguaggi regolari – FSA
 - Grammatiche non contestuali – linguaggi non contestuali – NPDA
 - Grammatiche generali – linguaggi ricorsivamente enumerabili – TM

Automati, linguaggi e grammatiche

Gerarchia di Chomsky	Grammatiche	Linguaggi	Automa minimo
Tipo 0	Generali	Ricorsivamente enumerabili	Macchina di Turing
Tipo 1	Dipendenti dal contesto	Dipendenti dal contesto	<i>Automa lineare limitato (LBA)</i>
Tipo 2	Non contestuali	Non contestuali	NPDA
Tipo 3	Regolari	Regolari	FSA

Uso dei tipi e note storiche

- Useremo principalmente i tipi 3 e 2
 - Definiremo il tipo 3 in modo molto più semplice
- Chomsky usò in seguito una numerazione dei tipi di grammatiche diversa da quella del 1959:
 - Tipo 1: grammatiche monotone o *noncontracting (NCG)*, cioè con produzioni $\alpha \rightarrow \beta$, con $|\alpha| \leq |\beta|$
 - Tipo 2: grammatiche dipendenti dal contesto o *context-sensitive (CSG)*
 - Benché sintatticamente diverse, NCG e CSG hanno lo stesso potere espressivo

Definizione

- Se per ogni $\alpha \rightarrow \beta \in P$ si ha $|\alpha| = 1$ e
 - $\beta \in V_N \cdot V_T \cup V_T \cup \{\epsilon\}$, la grammatica è **regolare sinistra**
- Se per ogni $\alpha \rightarrow \beta \in P$ si ha $|\alpha| = 1$ e
 - $\beta \in V_T \cdot V_N \cup V_T \cup \{\epsilon\}$, la grammatica è **regolare destra**
- Una grammatica è **regolare (RG)** sse è o regolare sinistra o regolare destra
- Un linguaggio è regolare sse è generato da una grammatica regolare
 - Ci dev'essere almeno una grammatica che lo genera

RG e FSA

Sia A un FSA. Si può costruire una RG G ad esso equivalente. Equivalente significa che G genera esattamente lo stesso linguaggio riconosciuto da A (e viceversa)

Le grammatiche regolari, gli automi a stati finiti e le espressioni regolari sono modelli diversi per descrivere la stessa classe di linguaggi

Costruzione di una RG da un FSA

- Se $A = \langle Q, I, \delta, q_0, F \rangle$, allora si può costruire $G = \langle V_N, V_T, P, S \rangle$ tale che
 - $V_N = Q$,
 - $V_T = I$,
 - $S = q_0$
 - Per ogni $\delta(q, i) = q'$
 - $q \rightarrow i q' \in P$
 - Se $q' \in F$ allora $q' \rightarrow \varepsilon \in P$
- $\delta^*(q, x) = q'$ se e solo se $q \Rightarrow^* xq'$

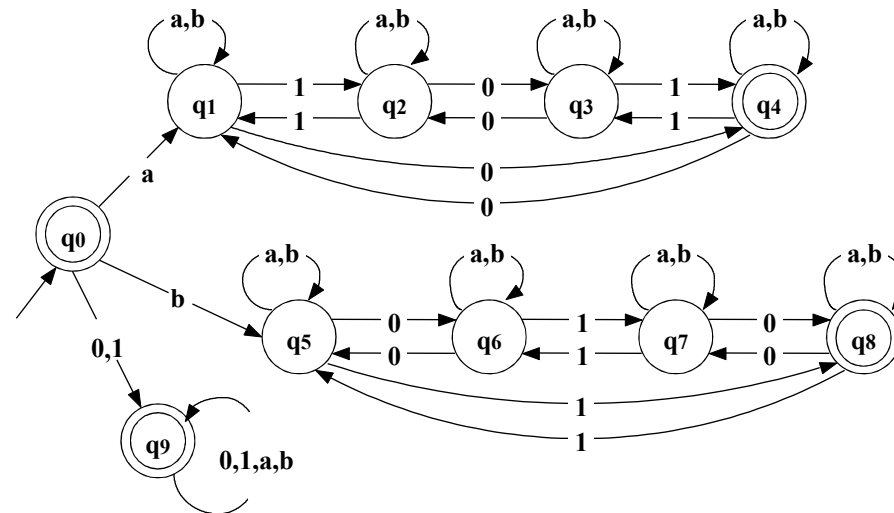
Costruzione di un FSA da una RG

Se $G = \langle V_N, V_T, P, S \rangle$ allora si può costruire $A = \langle Q, I, \delta, q_0, F \rangle$ tale che

- $Q = V_N \cup \{q_F\}$
- $I = V_T$,
- $q_0 = S$,
- $F = \{q_F\}$
- Per ogni $A \rightarrow bC$, $\delta(A, b) = C$
- Per ogni $A \rightarrow b$, $\delta(A, b) = q_F$

Esempio (1)

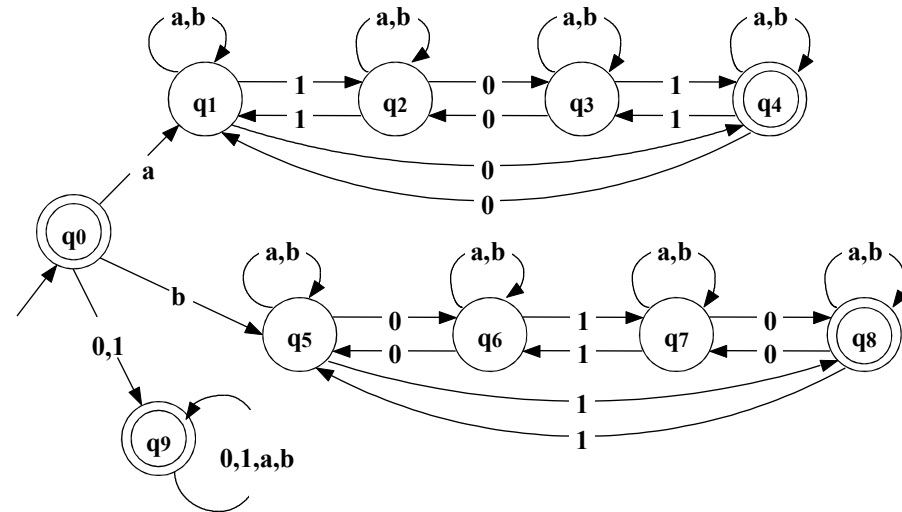
- Costruire un automa che riconosce il linguaggio L sull'alfabeto $\{a,b,0,1\}$, tale che una stringa x è in L se x ha le seguenti proprietà:
 - se x inizia con 'a' allora x ha un numero pari di '1' e un numero dispari di '0'
 - se x inizia con 'b' allora x ha un numero pari di '0' e un numero dispari di '1'



Esempio (2)

- Grammatica equivalente

- $S \rightarrow aS_1 \mid bS_5 \mid 0S_9 \mid 1S_9 \mid \epsilon$
- $S_1 \rightarrow aS_1 \mid bS_1 \mid 1S_2 \mid 0S_4$
- $S_2 \rightarrow aS_2 \mid bS_2 \mid 1S_1 \mid 0S_3$
- $S_3 \rightarrow aS_3 \mid bS_3 \mid 0S_2 \mid 1S_4$
- $S_4 \rightarrow aS_4 \mid bS_4 \mid 0S_1 \mid 1S_3 \mid \epsilon$
- $S_5 \rightarrow aS_5 \mid bS_5 \mid 0S_6 \mid 1S_8$
- $S_6 \rightarrow aS_6 \mid bS_6 \mid 0S_5 \mid 1S_7$
- $S_7 \rightarrow aS_7 \mid bS_7 \mid 0S_8 \mid 1S_6$
- $S_8 \rightarrow aS_8 \mid bS_8 \mid 0S_7 \mid 1S_5 \mid \epsilon$
- $S_9 \rightarrow aS_9 \mid bS_9 \mid 0S_9 \mid 1S_9 \mid \epsilon$



Definizione

- Una grammatica è detta **non contestuale** (CFG, cioè *context-free grammar*) se
 - per ogni $\alpha \rightarrow \beta \in P$, si ha $|\alpha| = 1$, cioè α è un elemento di V_N .
- Sono dette non contestuali perché la riscrittura di α non dipende dal contesto di α
 - contesto di α = parte della stringa che circonda α

Grammatiche non contestuali

- Le CFG sono la stessa cosa delle BNF (BNF = Backus-Naur Form) usate per definire la sintassi dei linguaggi di programmazione
 - sono adatte a definire le caratteristiche tipiche dei linguaggi di programmazione
- Le grammatiche regolari sono anche non contestuali
 - Ma non viceversa

Esempio di BNF

<if_statement> ::=

if <boolean_expression> **then**

 <statement_sequence>

 [**else** <statement_sequence>]

end if ;

<statement_sequence> ::= <statement> [;

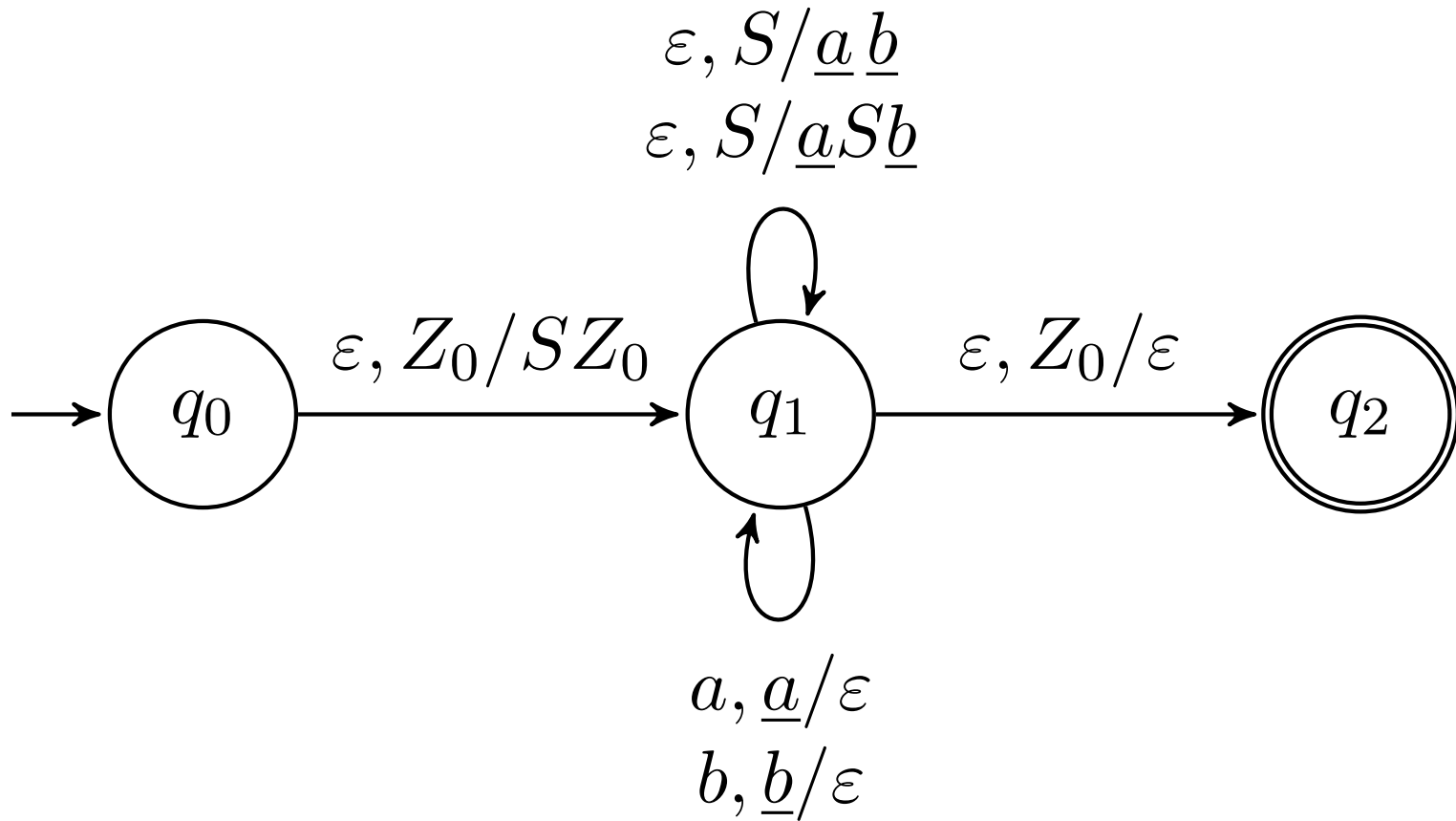
 <statement_sequence>]

- Terminali in **grassetto rosso**
- Non terminali delimitati da parentesi angolari
- Elementi opzionali tra quadre

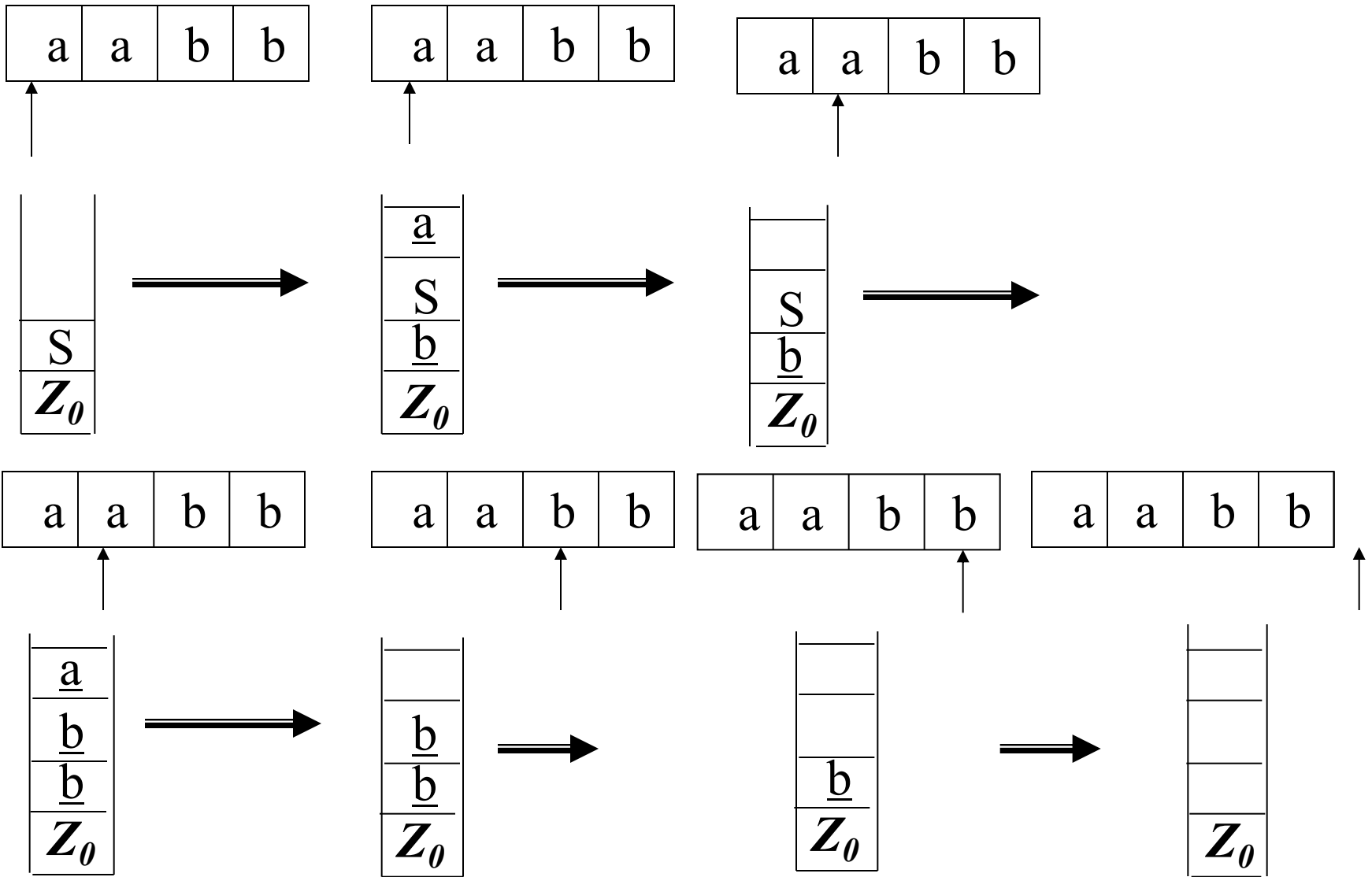
Le CFG sono equivalenti agli NPDA

Giustificazione intuitiva (senza dimostrazione: la dimostrazione è al centro della costruzione dei compilatori)

Consideriamo la grammatica $S \rightarrow aSb \mid ab$

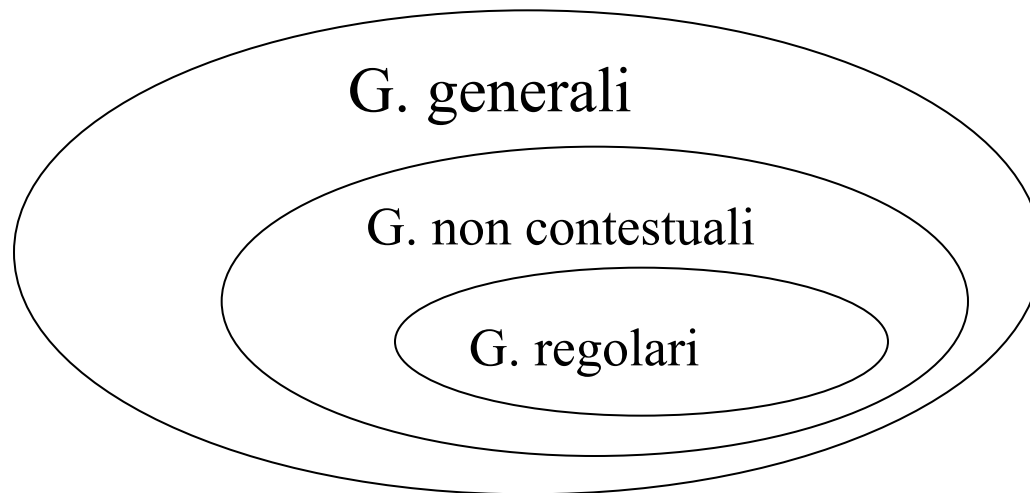


$$S \Rightarrow aSb \Rightarrow aabb$$



Definizione

- Le grammatiche generali (o non ristrette) sono grammatiche senza limitazioni sulle produzioni
 - Corrispondono al tipo 0 della gerarchia di Chomsky
- Sia le grammatiche non contestuali, sia le regolari sono (anche) non ristrette



Grammatiche generali e TM

- Le grammatiche generali (GG) e le TM sono formalismi equivalenti
 - Data una GG si può costruire una TM che riconosca il linguaggio generato dalla GG
 - Data una TM si può definire una GG che genera il linguaggio accettato dalla TM
- Come?

Da una GG a una TM (1)

Data una grammatica generale $G = \langle V_N, V_T, P, S \rangle$, costruiamo una NTM M tale che $L(M) = L(G)$:

- M ha un nastro di memoria
- La stringa di ingresso x è sul nastro di ingresso
- Il nastro di memoria è inizializzato con S (o meglio: Z_0S)
- Il nastro di memoria in generale conterrà una qualche stringa $\alpha \in V^*$
 - Viene scandito per cercare la parte sinistra di una produzione di P
 - Quando se ne trova una (non necessariamente la prima), M compie una scelta non deterministica e la parte scelta è sostituita dalla parte destra corrispondente (se ci sono diverse parti destre, ancora una volta, M compie una scelta non deterministica)

Da una GG a una TM (2)

- Così, ogni volta che $\alpha \Rightarrow \beta$, abbiamo $c_s = \langle q_s, Z_0\alpha \rangle \vdash^* \langle q_s, Z_0\beta \rangle$ per qualche stato q_s
- Se e quando il nastro contiene una stringa $y \in V_T^*$, è confrontata con x
 - Se coincidono, x è accettata
 - Altrimenti questa particolare sequenza di mosse non porta all'accettazione

Note

- Usare una NTM facilita la costruzione ma non è necessario
- Notare che, se $x \notin L(G)$, M può tentare infinite computazioni, nessuna delle quali porta ad accettazione
 - alcune di queste potrebbero non terminare mai, quindi (correttamente) senza poter concludere che $x \in L(G)$
 - *e anche* senza poter concludere che $x \notin L(G)$

Infatti la definizione di accettazione richiede che M raggiunga una configurazione di accettazione se e solo se $x \in L$

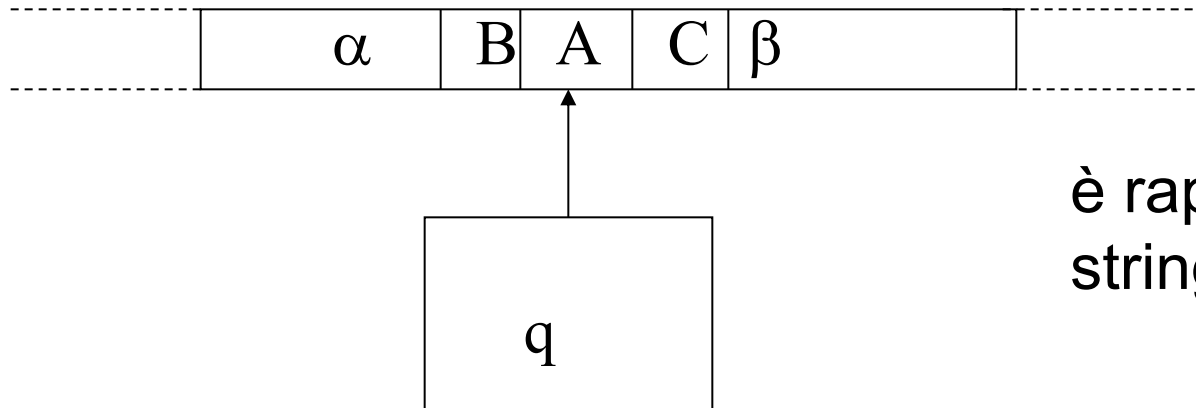
- Non richiede che M termini la computazione in uno stato non finale se $x \notin L$
- Ancora una volta abbiamo il problema del complemento e notiamo l'asimmetria tra la risoluzione di un problema in senso positivo o negativo

Da una TM a una GG (1)

Data una TM a nastro singolo M , costruiamo una grammatica generale G che genera $L(M)$:

- Prima G genera tutte le stringhe del tipo $x\$X$, $x \in V_T^*$, dove X è una “copia di x ” fatta di simboli non terminali (es., per $x = aba$, $x\$X = aba\ABA)
 - Questo si può fare con un numero finito di produzioni
- G simula le configurazioni di M usando la stringa a destra di $\$$
- G ha una derivazione $x\$X \Rightarrow^* x$ sse x è accettata da M
 - L’idea di base è di simulare ogni mossa di M con una derivazione immediata di G

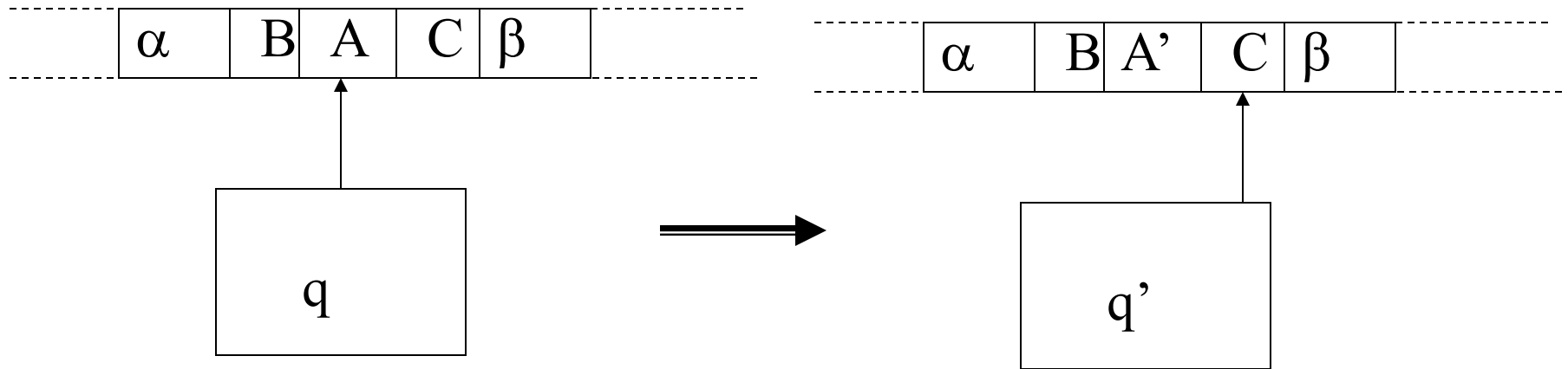
Da una TM a una GG (2)



è rappresentata dalla stringa $\alpha BqAC\beta$ dopo il \$

- G ha quindi derivazioni della forma $x\$X \Rightarrow x\q_0X (configurazione iniziale di M a destra del \$)
- Inoltre
 - Se, in M, $\delta(q,A) = \langle q', A', R \rangle$ allora G include la produzione $qA \rightarrow A'q'$
 - Se, in M, $\delta(q,A) = \langle q', A', S \rangle$ allora G include la produzione $qA \rightarrow q' A'$
 - Se, in M, $\delta(q,A) = \langle q', A', L \rangle$ allora G include la produzione $BqA \rightarrow q' BA' \forall B$ nell'alfabeto di M (si ricordi che M è a nastro singolo, quindi ha un unico alfabeto per ingresso, memoria e uscita)

Da una TM a una GG (3)



Se e solo se: $x\$ \alpha B q A C \beta \Rightarrow x\$ \alpha B A' q' C \beta$, ecc.

- Alla fine aggiungiamo produzioni che consentono a G di derivare da $x\$ \alpha B q_f A C \beta$ la sola stringa x sse M raggiunge una configurazione di accettazione ($\alpha B q_f A C \beta$), cancellando qualunque cosa si trovi a destra del $\$$ (incluso il $\$$)