

# Grammaticiche

Dipartimento di Elettronica, Informazione e Bioingegneria  
Politecnico di Milano

13 marzo 2024

# Modelli Generativi

## Grammatiche

- I modelli di linguaggio/calcolo visti finora definiscono un linguaggio tramite l'*elaborazione* della stringa che gli appartiene
- Vediamo un modello *generativo* del linguaggio: la grammatica
- In generale, una *grammatica* o *sintassi* è un insieme di regole per generare *frasi* di un linguaggio
- Modello molto antico (Pāṇini ne fornì una per il sanscrito tra il 6<sup>o</sup> e il 4<sup>o</sup> secolo a.C.), ma di grande efficacia

# Grammatica come sistema di riscritture

## Esempi (informali)

- “Una frase è formata da un soggetto seguito da un predicato”
  - “Un soggetto è un sostantivo, o un pronome, oppure ...”
  - “Un predicato è un verbo seguito da complementi, oppure ...”
  - Frase  $\rightarrow$  soggetto.predicato  $\rightarrow$  “Pierino mangia la mela”
  - Specifica sintattica: vale anche “La mela mangia Pierino”
- “Una funzione ANSI-C-89 è composta da un prototipo, una parte dichiarativa, una esecutiva”
  - Programma  $\rightarrow$  prototipo.dichiarativa.esecutiva  $\rightarrow$  `int f(void) {int a; a=a+1; return a;}`
- “Un cartello di segnaletica verticale è di obbligo, divieto, pericolo, precedenza,...”
  - “Un cartello di pericolo è di forma triangolare, bordato in rosso e contiene...”

# Grammatica come sistema di riscritture

## Riscritture per raffinamenti successivi

- Le regole di una grammatica descrivono un “oggetto principale” (libro, protocollo, messaggio grafico) come un insieme ordinato di “componenti”
- La descrizione è fornita fino ad arrivare al livello di dettaglio desiderato (carattere, bit, forma geometrica elementare)
- Ogni passo di riscrittura può offrire una o più alternative
  - Il soggetto può essere un nome, un pronome ...
- Spesso si tende a chiamare *lessico* la descrizione grammaticale delle singole “parole”, *sintassi* quella della loro composizione
  - Dal nostro punto di vista, è un riuso dello stesso modello

# Grammatica

## Definizione formale

- Una grammatica è una quadrupla  $G = \langle \mathbf{V}_t, \mathbf{V}_n, \mathbf{P}, S \rangle$ 
  - $\mathbf{V}_t$ : alfabeto o vocabolario *terminale*
  - $\mathbf{V}_n$ : alfabeto o vocabolario *nonterminale*
  - $\mathbf{V} = \mathbf{V}_n \cup \mathbf{V}_t$ : alfabeto o vocabolario
  - $S \in \mathbf{V}_n$ : elemento di  $\mathbf{V}_n$  detto *assioma* o *simbolo iniziale*
  - $\mathbf{P} \subseteq \mathbf{V}_n^+ \times \mathbf{V}^*$  insieme delle produzioni sintattiche o regole di riscrittura
- Per semplicità di notazione, indicheremo gli elementi  $p \in \mathbf{P}$ ,  $p = \langle \alpha, \beta \rangle$  come  $p = \alpha \rightarrow \beta$

# Grammatica

## La relazione di derivazione

- Definiamo la relazione di derivazione immediata  $\Rightarrow_G$  per una grammatica  $G = \langle \mathbf{V}_t, \mathbf{V}_n, \mathbf{P}, S \rangle$  come  $\alpha \Rightarrow_G \beta$  se e solo se  $\alpha \in V^+, \beta \in V^*, \alpha = \alpha_1 \alpha_2 \alpha_3, \beta = \alpha_1 \beta_2 \alpha_3, \alpha_2 \rightarrow \beta_2 \in \mathbf{P}$
- Esempio: Rispetto a  $G = \langle \{ab\}, \{S, R\}, \{S \rightarrow aR, R \rightarrow bS, S \rightarrow \varepsilon\}, S \rangle$ , abbiamo che  $abaR \Rightarrow_G ababS$  ma non è vero che  $abaR \Rightarrow_G aba$
- Dove non ambiguo, ometteremo il pedice  $G$  che indica la grammatica
- Definiamo  $\Rightarrow^*$  come la chiusura riflessiva e transitiva di  $\Rightarrow$

## Linguaggio generato da una grammatica

Il linguaggio  $L(G)$  generato dalla grammatica  $G$  è l'insieme di tutte e sole le stringhe  $x$  di soli caratteri di  $\mathbf{V}_t$  tali che  $S \Rightarrow^* x$ .

## Alcuni esempi - 1

### Una prima grammatica semplice $G_1$

- $V_t = \{a, b, c\}$ ,  $V_n = \{S, A, B, C\}$  con assioma  $S$
- $P = \{S \rightarrow A, A \rightarrow aA, A \rightarrow B, B \rightarrow bB, B \rightarrow C, C \rightarrow cC, C \rightarrow \varepsilon\}$
- Una possibile derivazione è  
 $S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow aaB \Rightarrow aaC \Rightarrow aacC \Rightarrow aaccC \Rightarrow aacccC \Rightarrow aaccc$
- Un'altra possibile derivazione è  $S \Rightarrow A \Rightarrow B \Rightarrow bB \Rightarrow bC \Rightarrow b$
- Linguaggio generato da  $G$ ,  $L(G) = \{a^*b^*c^*\}$

## Alcuni esempi - 2

### Qualcosa di più sostanzioso $G_2$

- $V_t = \{a, b\}$ ,  $V_n = \{S\}$  assioma  $S$
- $P = \{S \rightarrow aSbS, S \rightarrow \varepsilon\}$
- Una possibile derivazione  $S \Rightarrow aSbS \Rightarrow aSb \Rightarrow aaSbSb \Rightarrow aaSbb \Rightarrow aabb$
- Una ulteriore possibile derivazione  
 $S \Rightarrow aSbS \Rightarrow aSbaSbS \Rightarrow abaSbS \Rightarrow ababS \Rightarrow abab$
- Linguaggio generato dalla grammatica? Coppie di  $a$  e  $b$  “ben parentetizzate”



## Alcuni esempi - 3

### Qualcosa di *ancora* più sostanzioso $G_3$

- $V_t = \{a, b, c\}$ ,  $V_n = \{S, A, B, C, D\}$  assioma  $S$
- $P = \{S \rightarrow aACD, A \rightarrow aAC, A \rightarrow \varepsilon, B \rightarrow b, CD \rightarrow BDc, CB \rightarrow BC, D \rightarrow \varepsilon\}$
- Una possibile derivazione:  
$$\begin{aligned} S &\Rightarrow aACD \Rightarrow aaACCD \Rightarrow aaaACCCD \Rightarrow aaaACCCD \Rightarrow aaaCCCD \Rightarrow \\ &aaaCCBDc \Rightarrow aaaCCBDc \Rightarrow aaaCBCDc \Rightarrow aaaBCCDc \Rightarrow aaaBCBDcc \Rightarrow \\ &aaaBBCDcc \Rightarrow aaaBBBDccc \Rightarrow aaaBBBccc \xrightarrow{*} aaabbbccc \end{aligned}$$
- Linguaggio generato:  $L(G) = \{a^n b^n c^n\}$

## Alcune domande “naturali”

### Utilità pratica

- Dove è possibile utilizzare grammatiche in pratica?

### Espressività

- Quali linguaggi è possibile esprimere con una data grammatica?

### Relazione con i riconoscitori

- Che relazione sussiste tra i linguaggi generati dalle grammatiche e i linguaggi riconosciuti dagli automi?

# Usi pratici delle grammatiche

## Modello descrittivo

- Le grammatiche sono ampiamente usate come modello descrittivo di linguaggi di programmazione (C, Scheme, Pascal, ...) e descrizione dati (JSON)
  - Esiste per alcune di esse la possibilità di ottenere automaticamente l'automa riconoscitore del linguaggio generato

## Modello generativo

- Generazione automatizzata di input di test per programmi
- Sintesi di frasi in linguaggio "naturale"

# Espressività delle grammatiche

## Quali linguaggi è possibile esprimere

- Negli esempi precedenti abbiamo visto come sia possibile generare linguaggi che sappiamo essere riconosciuti, usando un automa a potenza minima
  - Da un FSA: è facile costruire un FSA det. a 3 stati che riconosce  $L(G_1)$
  - Da un PDA: il riconoscitore di  $L(G_2)$  è stato un esempio che abbiamo visto durante le precedenti lezioni
  - Da una MT: serve una MT per riconoscere  $L(G_3) = a^n b^n c^n$
- É possibile classificare le grammatiche in base al loro potere generativo
  - ovvero in base alla famiglia di appartenenza del linguaggio generato

# Espressività delle grammatiche

## Quali linguaggi è possibile esprimere? La gerarchia di Chomsky

- 4 classi, a seconda delle limitazioni (crescenti) imposte sulle produzioni  $\alpha \rightarrow \beta$

Tipo	Nome	Limitazione Produzioni
0	Non limitate	nessuna
1	Monotone (non-decreasing)	$ \alpha  \leq  \beta $
1	Dipendenti dal contesto	$\alpha = \gamma\chi\delta, \beta = \gamma\chi\delta, \chi \neq \varepsilon$ o $S \rightarrow \varepsilon$
2	Libere dal contesto	$ \alpha  = 1$
3	Regolari	di forma $A \rightarrow a, A \rightarrow aA$ , oppure $A \rightarrow a, A \rightarrow Aa$ con $a \in \mathbf{V}_t, A \in \mathbf{V}_n$

- Una grammatica più potente genera tutti i linguaggi di una meno potente.  
Domanda: l'inclusione è stretta?

# A quali automi corrispondono?

## Grammatiche Regolari e FSA

- Linguaggi generati da grammatiche regolari  $\equiv$  riconosciuti da FSA

## Dall'FSA $\mathcal{A}$ alla grammatica

- Poniamo  $\mathbf{V}_n = \mathbf{Q}$ ,  $\mathbf{V}_t = \mathbf{I}$ ,  $S = \langle q_0 \rangle$
- Per ogni  $\delta(q, i) = q'$  aggiungiamo  $\langle q \rangle \rightarrow i \langle q' \rangle$  all'insieme  $\mathbf{P}$
- Se  $q' \in \mathbf{F}$  per una data  $\delta(q, i) = q'$ , aggiungiamo anche  $\langle q \rangle \rightarrow i$  all'insieme  $\mathbf{P}$
- Facile mostrare per induzione che  $\delta^*(q_0, x) = q'$  sse  $\langle q_0 \rangle \xRightarrow{*} x \langle q' \rangle$

## Dalla grammatica all'FSA (non deterministico)

- $\mathbf{Q} = \mathbf{V}_n \cup \{q_f\}$ ,  $\mathbf{I} = \mathbf{V}_t$ ,  $q_0 = S$ ,  $\mathbf{F} = \{q_f\}$
- Se  $A \rightarrow bC \in \mathbf{P}$ ,  $\delta(A, b) = C$ ; Se  $A \rightarrow b \in \mathbf{P}$ ,  $\delta(A, b) = q_f$

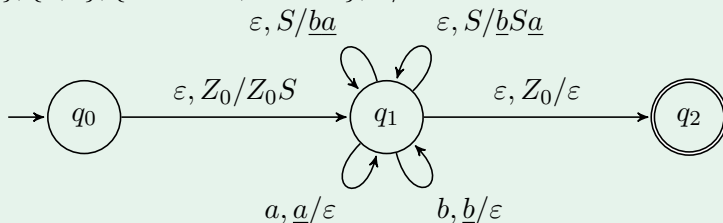
# A quali automi corrispondono?

## Grammatiche Libere dal Contesto e AP-ND

- I linguaggi generati dalle grammatiche libere dal contesto coincidono con i riconosciuti dagli AP non deterministici
- Dimostrazione non banale; intuizione: salvo in pila la coda della forma di frase

## Dalla grammatica all'AP ND: un esempio illustrativo

- Data  $\langle \{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow ab\}, S \rangle$



## A quali automi corrispondono?

### Grammatiche non limitate e MT

- Le grammatiche non limitate (tipo 0) corrispondono alle MT
- Costruiamo, senza pretesa di formalizzazione qui, una MT ND che accetti  $L(G)$
- $\mathcal{M}$  ha un nastro di memoria, inizializzato con  $Z_0S$
- La stringa da riconoscere è sul nastro di ingresso
- Il nastro di memoria viene scandito alla ricerca di una parte sinistra di una qualche produzione  $p \in \mathbf{P}$
- Quando una viene trovata (scelta nondeterministicamente), viene sostituita con la sua parte destra
- Se ve n'è più di una, si opera ancora nondeterministicamente



# Grammatiche non limitate e MT

## Funzionamento della MT

- Per come abbiamo costruito la MT ND, sappiamo che

$$\alpha \Rightarrow \beta \text{ se e solo se } c = \langle q, Z_0, \alpha \rangle \vdash^* \langle q, Z_0, \beta \rangle$$

- In questo modo, quando (e se) sul nastro di memoria si raggiunge un contenuto fatto di soli elementi di  $\mathbf{V}_t$ , lo si può confrontare con la stringa in ingresso  $x$  e
  - se coincide accettare  $x$
  - se non coincide, questa computazione tra quelle eseguite nondeterministicamente non è di accettazione
- N.B. Il nondeterminismo della MT è utile, ma non essenziale
- Resta il problema di sapere se la MT termina...

# Grammatiche non limitate e MT

## Emulare una MT con una grammatica non ristretta

- Senza perdere di generalità, emuliamo una MT  $\mathcal{M}$  a nastro singolo con una grammatica  $G$  non ristretta
- Considerato che  $G$  può “manipolare” solo elementi di  $V_n$ , faccio in modo che generi stringhe della forma  $x\blacklozenge X$  con  $\blacklozenge \in V_n, x \in V_t^*$  e  $X$  che è costituita da “copie nonterminali” degli elementi di  $X$ 
  - Ad esempio, se  $x = abac$ , genero la stringa  $abac\blacklozenge ABAC$
- Obiettivo: avere una derivazione  $x\blacklozenge X \xRightarrow{*} x$  se e solo se  $x$  è accettata da  $\mathcal{M}$
- Simuleremo ogni mossa di  $\mathcal{M}$  con una derivazione diretta di  $G$

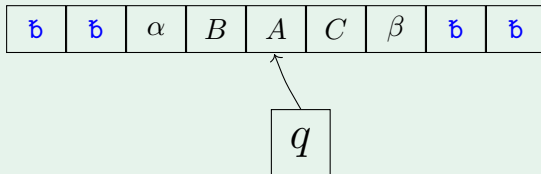
# Grammatiche non limitate e MT

## Emulazione della impostazione del nastro con parte iniziale di $G$

- Assumendo  $\mathbf{I} = \mathbf{V}_t = \{a, b\}$  la parte delle produzioni di  $G$  che genera le stringhe appena descritte è:
  - $S \rightarrow SA'A, S \rightarrow SB'B, S \rightarrow \blacklozenge$  (genero coppie di simboli)
  - $AA' \rightarrow A'A, BA' \rightarrow A'B$  (faccio “scorrere” le  $A'$  a sx)
  - $AB' \rightarrow B'A, BB' \rightarrow B'B$  (faccio “scorrere” le  $B'$  a sx)
  - $\blacklozenge A' \rightarrow a\blacklozenge, \blacklozenge B' \rightarrow b\blacklozenge$  (quando “scorro” attraverso  $\blacklozenge$  trasformo il nonterminale in terminale)

# Grammatiche non limitate e MT

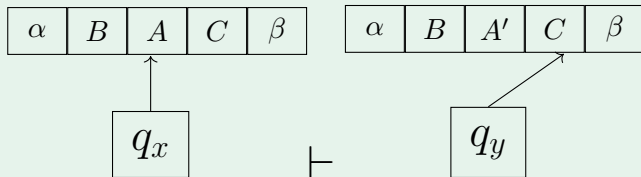
## Emulare le mosse



- Rappresento la configurazione qui sopra con  $\blacklozenge \alpha B Q A C \beta$
- Se è definita:
  - $\delta(q, A) = \langle q', A', R \rangle$  aggiungo  $QA \rightarrow A'Q'$  alle prod. di  $G$
  - $\delta(q, A) = \langle q', A', S \rangle$  aggiungo  $QA \rightarrow Q'A'$  alle prod. di  $G$
  - $\delta(q, A) = \langle q', A', L \rangle$  aggiungo,  $\forall B$  nell'alfabeto di  $\mathcal{M}$ ,  $BQA \rightarrow Q'BA'$  alle prod. di  $G$  (n.b. l'alfabeto di  $\mathcal{M}$  è unico)

## Grammatiche non limitate e MT

### Emulare le mosse



- se e solo se  $\blacklozenge \alpha B Q_x A C \beta \Rightarrow \blacklozenge \alpha B A' Q_y C \beta$
- La costruzione di  $G$  va completata aggiungendo regole che cancellano tutto ciò che sta a destra del  $\blacklozenge$  ( $\blacklozenge$  incluso) se e solo se la configurazione della  $\mathcal{M}$  è accettante, e.g.  $\blacklozenge \alpha B Q_f A C \beta$

# Rimanenti corrispondenze

## Automi a pila deterministici

- Esiste un sottoinsieme (proprio) delle grammatiche libere dal contesto che genera i linguaggi riconosciuti dagli AP deterministici
- Restrizione difficile da esprimere sulla forma delle produzioni
- Dettagliata nel corso di Formal Languages and Compilers

## Grammatiche dipendenti dal contesto

- Le grammatiche di tipo 1 corrispondono a un sottoinsieme delle MT di cui è certo che terminino sempre
- N.B. non si tratta delle uniche MT che terminano sempre
- Consentono sempre di sapere se una stringa  $x$  è generata da  $G$  (si può costruire l'insieme delle stringhe gen. da  $G$  lunghe quanto  $x$  e vedere se essa appare)