

Fondamenti di Informatica
per la Sicurezza
a.a. 2007/08

Grammatiche

Stefano Ferrari



UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI TECNOLOGIE DELL'INFORMAZIONE

Stefano Ferrari ★ Università degli Studi di Milano

Fondamenti di Informatica per la Sicurezza ◇ Grammatiche ◇ a.a. 2007/08 - p. 1/37

Alfabeto

Un **alfabeto** è un insieme **finito** e non vuoto di simboli.

Esempi:

- $A = \{a, b, c\}$
- $A = \{0, 1\}$

Stefano Ferrari ★ Università degli Studi di Milano

Fondamenti di Informatica per la Sicurezza ◇ Grammatiche ◇ a.a. 2007/08 - p. 2/37

Stringa

Una **stringa** (o **parola**) è una sequenza finita di simboli.

Esempio:

- se $A = \{a, b, c\}$, $aaabacab$ è una stringa su A .

La **lunghezza** di una stringa w , denotata con $|w|$, è il numero di simboli che la compongono.

Esempio:

- $|aaabacab| = 8$

La **stringa vuota** ϵ è composta da zero simboli:

$$|\epsilon| = 0$$

Sottostringhe

Sia $w = a_1 \cdots a_n$:

- $a_1 \cdots a_j$, con $j \in \{1, \dots, n\}$ è un **prefisso** di w ;
- $a_j \cdots a_n$, con $j \in \{1, \dots, n\}$ è un **suffisso** di w ;
- $a_i \cdots a_j$, con $i, j \in \{1, \dots, n\}$ è una **sottostringa** di w ;
- ϵ è prefisso, suffisso e sottostringa di w .

Concatenazione

La **concatenazione** di v e w è vw :

- la concatenazione è un'operazione associativa;
- ϵ ne è l'elemento neutro.

La concatenazione si estende agli alfabeti.

Per esempio, se $A = \{a, b, c\}$:

- $A^0 = \{\epsilon\}$
- $A^1 = \{a, b, c\}$
- $A^2 = \{aa, ab, ac, \dots, cc\}$
- $A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \cup \dots = \bigcup_i A^i$ (**chiusura** di A)

Linguaggio

Un **linguaggio (formale)** L definito su un alfabeto A è un sottoinsieme di A^* .

Una **frase** di un linguaggio è una stringa appartenente al linguaggio stesso.

Esempi

Esempi:

- il linguaggio delle frasi in italiano:
 - alfabeto: $A = \{a, b, \dots, z, \langle \text{spazio} \rangle\}$
 - stringhe su A : *cosa, il porto, ghsde afe li*
 - frasi: *la gatta corre*
- il linguaggio delle espressioni aritmetiche:
 - alfabeto: $A = \{0, \dots, 9, +, -, :, \times, (,)\}$
 - stringhe su A : *980, 34 + 61, : ()345+*
 - frasi: *(25 + 12) : 3*

Operazioni sui linguaggi (1)

Se L , L_1 e L_2 sono linguaggi su Σ , si possono definire le seguenti operazioni:

unione , $L_1 \cup L_2$:

- $L_1 \cup L_2 = \{w \mid w \in L_1 \vee w \in L_2\}$
- $|L_1 \cup L_2| \leq |L_1| + |L_2|$

intersezione , $L_1 \cap L_2$:

- $L_1 \cap L_2 = \{w \mid w \in L_1 \wedge w \in L_2\}$
- $|L_1 \cap L_2| \leq \min(|L_1|, |L_2|)$

Operazioni sui linguaggi (2)

differenza , $L_1 - L_2$:

- $L_1 - L_2 = \{w \mid w \in L_1 \wedge w \notin L_2\}$
- $|L_1 - L_2| \leq |L_1|$

complemento , \bar{L} :

- $\Sigma^* - L$
- $|\bar{L}| \leq \infty$

Operazioni sui linguaggi (3)

concatenazione , $L_1 L_2$:

- $L_1 L_2 = \{vw \mid v \in L_1, w \in L_2\}$
- $|L_1 L_2| \leq |L_1| \cdot |L_2|$

potenza , L^n :

- $L^0 = \{\epsilon\}$
- $L^1 = L$
- $L^k = \{v_1 v_2 \dots v_k \mid v_1, v_2, \dots, v_k \in L\}$
- $|L^k| \leq |L|^k$

Operazioni sui linguaggi (4)

chiusura (di Kleene) , L^* :

- $L^* = \cup_i L^i$
- se $L = \{\epsilon\}$, $|L^*| = 1$
- altrimenti, $|L^*| = \infty$

chiusura positiva , L^+ :

- $L^+ = \cup_i L^i, i > 0$
- $L^* = L^+ \cup \{\epsilon\}$
- $|L^+| = \infty$

Grammatica

Una **grammatica (formale)** definisce in modo rigoroso un linguaggio su un alfabeto, Σ .

Tecnicamente è una quadrupla $\langle \Sigma, V, P, S \rangle$:

- Σ , insieme finito di simboli terminali:
 - può anche includere la stringa vuota, ϵ ;
- V , insieme finito di simboli non terminali:
 - sono meta-simboli di appoggio ($V \cap \Sigma = \emptyset$);
 - rappresentano categorie sintattiche;
- P , insieme di regole di riscrittura del tipo $\alpha \rightarrow \beta$, con α, β stringhe su $\Sigma \cup V$, con $\alpha \neq \epsilon$;
- $S \in V$, simbolo iniziale, detto **scopo**.

Generazione del linguaggio

Per ottenere una frase del linguaggio:

1. si parte dallo scopo, S ;
2. si applica una regola di produzione $\alpha \rightarrow \beta$:
 - (a) si cerca la presenza della sotto-sequenza α ;
 - (b) la si sostituisce con β ;
 - (c) α e β sono chiamate **forme di frase**;
3. si procede finché non si ottiene una stringa con soli simboli terminali.

Tutte le frasi del linguaggio si ottengono con questo procedimento.

Esempio

Grammatica: $\langle \Sigma, V, P, S \rangle$

- $\Sigma = \{0, 1\}$
- $V = \{X\}$
- $P = \{X \rightarrow 0, X \rightarrow 1X, X \rightarrow 0X\}$
- $S = X$

Linguaggio: $\{0, 10, \dots, 0110, \dots\}$

- $S = X \rightarrow 0$
- $S = X \rightarrow 1X \rightarrow 10$
- $S = X \rightarrow 0X \rightarrow 01X \rightarrow 011X \rightarrow 0110$

Grammatiche BNF

La **notazione BNF** (Backus-Naur Form) è più conveniente per rappresentare le grammatiche:

- le regole di produzione sono della forma $\alpha ::= \beta$;
- i meta-simboli in V sono della forma $\langle nome \rangle$;
- il meta-simbolo speciale $|$ (pipe) è usato per l'alternativa:
 - $\alpha ::= \beta_1, \alpha ::= \beta_2, \dots, \alpha ::= \beta_n$;
 - $\alpha ::= \beta_1 | \beta_2 | \dots | \beta_n$.

Su <http://www.faqs.org/rfcs/rfc2234.html> si trova la definizione di Augmented BNF (ABNF).

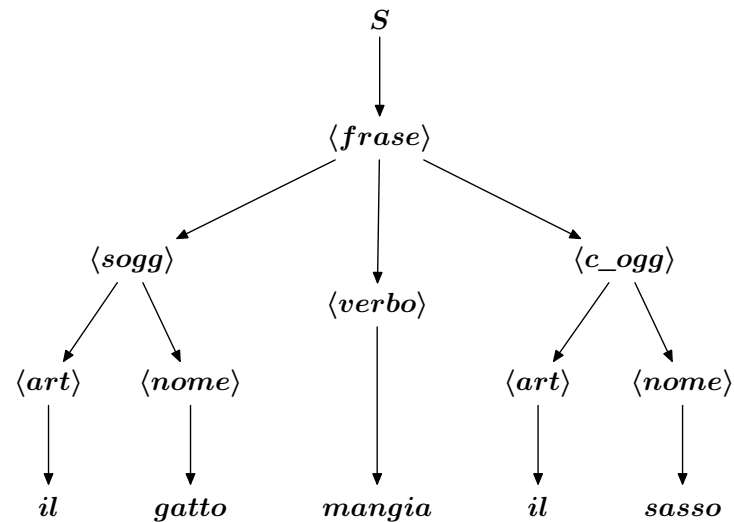
Esempio

Grammatica: $\langle \Sigma, V, P, S \rangle$

- $\Sigma = \{il, gatto, topo, sasso, mangia, beve\}$
- $V = \{\langle frase \rangle, \langle sogg \rangle, \langle art \rangle, \langle nome \rangle, \langle verbo \rangle, \langle c_ogg \rangle\}$
- $P = \{$
 - $\langle frase \rangle ::= \langle sogg \rangle \langle verbo \rangle \langle c_ogg \rangle$
 - $\langle sogg \rangle ::= \langle art \rangle \langle nome \rangle$
 - $\langle art \rangle ::= il$
 - $\langle nome \rangle ::= gatto | topo | sasso$
 - $\langle verbo \rangle ::= mangia | beve$
 - $\langle c_ogg \rangle ::= \langle art \rangle \langle nome \rangle \}$
- $S = \langle frase \rangle$

Albero sintattico

Esempio: *il gatto mangia il sasso*



Classificazione di Chomsky

È possibile caratterizzare le grammatiche sulla base della forma delle loro regole di produzione:

- Tipo 3 (**regolari**);
- Tipo 2 (**context-free**);
- Tipo 1 (**context-sensitive**);
- Tipo 0 (**a struttura di frase**).

Grammatiche di tipo 3

Le grammatiche regolari si dividono in:

- **lineari a destra**, $F ::= \alpha, F ::= \alpha G$;
- **lineari a sinistra**, $F ::= \alpha, F ::= G\alpha$;

con $F, G \in V, \alpha \in \Sigma^*$.

Esempi:

- $F ::= aG|a, G ::= bG|b|aF$
- $F ::= Fa|a$

Il linguaggio generato si dice di tipo 3 (o **linguaggio regolare**).

Grammatiche di tipo 2

Le grammatiche context-free (libere dal contesto) hanno produzioni della forma:

$F ::= \alpha$, con $F \in V$ e $\alpha \in (\Sigma \cup V)^*$.

Esempi:

- $F ::= aG|a, G ::= Gb|b|Fc$
- $F ::= aFc|b$

Si dice che il linguaggio è di tipo 2 (o **linguaggio libero dal contesto**).

Tipicamente, i linguaggi di programmazione appartengono a questo tipo.

Grammatiche di tipo 1

Le grammatiche context-sensitive (dipendenti dal contesto) hanno produzioni della forma:

- $\alpha F \beta ::= \alpha \gamma \beta$, $\alpha, \beta, \gamma \in (\Sigma \cup V)^*$, $\gamma \neq \epsilon$, $F \in V$;
- $S ::= \epsilon$.

Note:

- α e β definiscono il "contesto" in cui la sostituzione $F ::= \gamma$ può avvenire;
- per una produzione della prima forma, la lunghezza della stringa generata non può diminuire.

Esempio: $aFb ::= aFbGb$

Grammatiche di tipo 0

Le grammatiche di tipo 0 ammettono produzioni di qualsiasi forma:

- $\alpha ::= \beta$, con $\alpha, \beta \in (\Sigma \cup V)^*$.

Nota: sono ammesse anche produzioni che possono diminuire la lunghezza della stringa generata.

Gerarchia di grammatiche e linguaggi

Ogni grammatica di tipo n è anche una grammatica di tipo $n - 1$.

Un linguaggio è di tipo n se esiste una grammatica di tipo n che lo genera, ma non nessuna di tipo $n + 1$ è in grado di generarlo.

Esempi:

- linguaggio di tipo 3: $\{a^m b^n \mid m, n \geq 0\}$
- linguaggio di tipo 2: $\{a^n b^n \mid n \geq 0\}$
- linguaggio di tipo 1: $\{a^n b^n c^n \mid n \geq 0\}$
- linguaggio di tipo 0: $\{a^n \mid n \text{ è un numero primo}\}$

Grammatiche ambigue

Una grammatica si dice **ambigua** se qualche stringa del linguaggio da essa generato può essere generata mediante più alberi sintattici.

Esempio: con la regola $F ::= FaF \mid FbF \mid f$, la generazione della stringa $fafbf$ non è univoca.

Infatti:

- $F \rightarrow FaF \rightarrow faF \rightarrow faFbF \rightarrow fafbF \rightarrow fafbf$
- $F \rightarrow FbF \rightarrow Fbf \rightarrow FaFbf \rightarrow faFbf \rightarrow fafbf$

Un linguaggio si dice **inerentemente ambiguo** se non esiste alcuna grammatica non ambigua che lo generi.

Esercizio 1 (1)

Definiti i linguaggi:

- $L_1 = \{aa, ab, bc, c\}$
- $L_2 = \{1, 22, 31\}$

descrivere i linguaggi:

- | | |
|----------------------|-----------------------------|
| a) $L_3 = L_1^*$ | d) $L_6 = L_1 \cup L_2$ |
| b) $L_4 = L_1 L_2$ | e) $L_7 = (L_1 L_2)^*$ |
| c) $L_5 = L_1 L_2^*$ | f) $L_8 = (L_1 \cup L_2)^*$ |

Esercizio 1 (2)

- a) $L_3 = L_1^*$ è formato dalla concatenazione di un numero arbitrario (anche nullo) di elementi di L_1 .

$$L_3 = \{\epsilon, aa, c, cababc, \dots\}$$

- b) $L_4 = L_1 L_2$ è formato dalla concatenazione di un elemento di L_1 con un elemento di L_2 .

$$L_4 = \{aa1, ab1, bc1, c1, aa22, ab22, bc22, c22, aa31, ab31, bc31, c31\}$$

- c) $L_5 = L_1 L_2^*$ è formato dalla concatenazione di un elemento di L_1 con un numero arbitrario (anche nullo) di elementi di L_2 .

$$L_5 = \{aa, \dots, c, bc1223131, ab111312222, \dots\}$$

Esercizio 1 (3)

d) $L_6 = L_1 \cup L_2$ è l'unione di L_1 e L_2 .

$$L_6 = \{aa, ab, bc, c, 1, 22, 31\}$$

e) $L_7 = (L_1 L_2)^*$ è la chiusura di L_4 .

È pertanto formato da una sequenza (eventualmente vuota) di elementi di L_4 .

$$L_7 = \{\epsilon, aa22c1, ab1aa1, \dots\}$$

f) $L_8 = (L_1 \cup L_2)^*$ è la chiusura di L_6 .

È pertanto formato da una sequenza (eventualmente vuota) di elementi di L_6 .

$$L_8 = \{\epsilon, c1122aacbc, 22131311aa, aa, 1, \dots\}$$

Esercizio 2 (1)

Data la grammatica $G = \langle \Sigma, V, P, S \rangle$:

- $\Sigma = \{a, b\}$
- $V = \{S, X, Y\}$
- $P = \{S ::= Y|a, Y ::= bX|aX, X ::= aY|b\}$

mostrare la sequenza di regole da applicare per generare le seguenti frasi:

a) bb

b) $aababb$

c) $babaab$

Esercizio 2 (2)

a) bb $S ::= Y|a, Y ::= bX|aX, X ::= aY|b$

bb	
$S ::= Y$	S
$Y ::= bX$	Y
$X ::= b$	bX
	bb

Esercizio 2 (3)

b) $aababb$ $S ::= Y|a, Y ::= bX|aX, X ::= aY|b$

$aababb$	
$S ::= Y$	S
$Y ::= aX$	Y
$X ::= aY$	aX
$Y ::= bX$	aaY
$X ::= aY$	$aabX$
$Y ::= bX$	$aabaY$
$X ::= b$	$aababX$
	$aababb$

Esercizio 2 (4)c) *babaab* $S ::= Y|a, Y ::= bX|aX, X ::= aY|b$

<i>babaab</i>	
	<i>S</i>
$S ::= Y$	<i>Y</i>
$Y ::= bX$	<i>bX</i>
$X ::= aY$	<i>baY</i>
$Y ::= bX$	<i>babX</i>
$X ::= aY$	<i>babaY</i>
$Y ::= aX$	<i>babaaX</i>
$X ::= b$	<i>babaab</i>

Esercizio 3 (1)Data la grammatica $G = \langle \Sigma, V, P, S \rangle$:

- $\Sigma = \{a, b, c\}$
- $V = \{S, X, Y\}$
- $P = \{S ::= X|aX, X ::= Xb|bXY|b, Y ::= cY|c\}$

mostrare la sequenza di regole da applicare per generare le seguenti frasi:

a) *bbb*b) *abbcc*c) *abbbcc*

Esercizio 3 (2)

a) bbb $S ::= X|aX, X ::= Xb|bXY|b, Y ::= cY|c$

bbb	
	S
$S ::= X$	X
$X ::= Xb$	Xb
$X ::= Xb$	Xbb
$X ::= b$	bbb

Esercizio 3 (3)

b) $abbcc$ $S ::= X|aX, X ::= Xb|bXY|b, Y ::= cY|c$

$abbcc$	
	S
$S ::= aX$	aX
$X ::= bXY$	$abXY$
$X ::= b$	$abbY$
$Y ::= cY$	$abbcY$
$Y ::= c$	$abbcc$

Esercizio 3 (4)

c) $abbbcc$ $S ::= X|aX$, $X ::= Xb|bXY|b$, $Y ::= cY|c$

$abbbcc$		$abbbcc$	
	S		S
$S ::= aX$	aX	$S ::= aX$	aX
$X ::= bXY$	$abXY$	$X ::= bXY$	$abXY$
$X ::= bXY$	$abbXY$	$X ::= Xb$	$abXbY$
$X ::= b$	$abbbY$	$X ::= b$	$abbbY$
$Y ::= c$	$abbbcY$	$Y ::= cY$	$abbbcY$
$Y ::= c$	$abbbcc$	$Y ::= c$	$abbbcc$

Ambigua!

Esercizio 4 (1)

Data la grammatica $G = \langle \Sigma, V, P, S \rangle$:

- $\Sigma = \{a, b, c\}$
- $V = \{S, X, Y\}$
- $P = \{S ::= X|aX, X ::= aXb|b|XY, Y ::= b|XYa|c\}$

mostrare la sequenza di regole da applicare per generare le seguenti frasi:

a) $aabbcbb$

b) $aabbbca$

Esercizio 4 (2)

a)

aabbcb

	<i>S</i>
<i>S</i> ::= <i>X</i>	<i>X</i>
<i>X</i> ::= <i>aXb</i>	<i>aXb</i>
<i>X</i> ::= <i>aXb</i>	<i>aaXbb</i>
<i>X</i> ::= <i>XY</i>	<i>aaXYbb</i>
<i>Y</i> ::= <i>c</i>	<i>aaXcbb</i>
<i>X</i> ::= <i>XY</i>	<i>aaXYcbb</i>
<i>X</i> ::= <i>b</i>	<i>aabYcbb</i>
<i>Y</i> ::= <i>b</i>	<i>aabbcb</i>

b)

aabbbca

	<i>S</i>
<i>S</i> ::= <i>aX</i>	<i>aX</i>
<i>X</i> ::= <i>XY</i>	<i>aXY</i>
<i>Y</i> ::= <i>XYa</i>	<i>aXXXYa</i>
<i>Y</i> ::= <i>c</i>	<i>aXXXca</i>
<i>X</i> ::= <i>b</i>	<i>aXbca</i>
<i>X</i> ::= <i>aXb</i>	<i>aaXbbca</i>
<i>X</i> ::= <i>b</i>	<i>aabbbca</i>