

# Corso di Linguaggi di Programmazione + Laboratorio Docente: Marco de Gemmis

## Capitolo 2 – Grammatiche e Linguaggi

Si ringraziano il Prof. Giovanni Semeraro e il Dott. Pasquale Lops per la concessione del materiale didattico

# Linguaggi formali e monoidi liberi

- Il concetto di **linguaggio formale** è strettamente correlato a quello di *monoide libero* (generato da un insieme).

# Definizione di Alfabeto

- Un insieme  $X$  finito e non vuoto di simboli è un *alfabeto*.
- Esempi
  - L'alfabeto latino, con l'aggiunta dei simboli di interpunzione e dello spazio bianco:  $a b c \dots z ; , . :$
  - L'insieme delle dieci cifre arabe:  $0 1 \dots 9$
- Con i simboli primitivi dell'alfabeto si formano le parole (es.:  $abc, 127, casa, \dots$ ).

# Definizione di Parola o Stringa

- Una sequenza finita di simboli  $x_1x_2\dots x_n$ , dove ogni  $x_i$  è preso da uno stesso alfabeto  $X$  è una *parola* (su  $X$ ).
- Esempio
  - $X = \{0,1\}$ .
  - 001110 è una parola su  $X$
- Una parola è ottenuta giustapponendo o concatenando simboli (caratteri) dell'alfabeto.
- Se una stringa ha  $m$  simboli (non necessariamente distinti) allora diciamo che ha lunghezza  $m$ .

# Lunghezza di una Parola o Stringa

- La lunghezza di una stringa  $w$  è denotata con  $|w|$ .
- Le parole di lunghezza 1 sono i simboli di  $X$ .
  - Quindi 001110 è una parola di lunghezza 6

$$|001110| = 6$$

- La parola vuota (o stringa vuota), denotata con  $\lambda$ , è una stringa priva di simboli ed ha lunghezza 0

$$|\lambda| = 0$$

# Definizioni

- **Uguaglianza tra stringhe**
  - Due stringhe sono *uguali* se i loro caratteri, letti ordinatamente da sinistra a destra, coincidono.
- $X^*$ 
  - L'insieme di tutte le stringhe di lunghezza finita sull'alfabeto  $X$  si denota con  $X^*$ .
- **Esempio**
  - Se  $X = \{0,1\}$ , allora  $X^* = \{\lambda, 0, 1, 00, 01, 10, 11, \dots\}$
- $X^*$  ha un numero di elementi che è un infinito numerabile.
  - Dalla definizione, segue che  $\lambda \in X^*$ , per ogni insieme  $X$ .

# Definizioni

## ■ Concatenazione o prodotto

- Sia  $\alpha \in X^*$  una stringa di lunghezza  $m$  e  $\beta \in X^*$  una stringa di lunghezza  $n$ , la **concatenazione** di  $\alpha$  e  $\beta$ , denotata con  $\alpha\beta$  o  $\alpha \cdot \beta$ , è definita come la stringa di lunghezza  $m+n$ , i cui primi  $m$  simboli costituiscono una stringa uguale a  $\alpha$  ed i cui ultimi  $n$  simboli costituiscono una stringa uguale a  $\beta$ .
- Quindi se  $\alpha = x_1x_2\dots x_m$  e  $\beta = x'_1x'_2\dots x'_n$ , si ha:

$$\alpha\beta = x_1x_2\dots x_mx'_1x'_2\dots x'_n$$

- Se  $X =$  alfabeto latino

$$\alpha = \text{capo} \quad \beta = \text{stazione} \quad \alpha\beta = \text{capostazione}$$

# Operazione di concatenazione

- La concatenazione di stringhe su  $X$  è una operazione binaria su  $X^*$ :

$$\cdot : X^* \times X^* \rightarrow X^*$$

- è **associativa**:  $(\alpha \beta )\gamma = \alpha (\beta \gamma ) = \alpha \beta \gamma , \quad \forall \alpha , \beta , \gamma \in X^*$
  - **non è commutativa**:  $\exists \alpha , \beta \in X^* : \alpha \beta \neq \beta \alpha$ 
    - *capostazione  $\neq$  stazionecapo*
  - ha **elemento neutro**  $\lambda$ :  $\lambda \alpha = \alpha \lambda = \alpha , \quad \forall \alpha \in X^*$
- Dunque  $(X^* , \cdot )$  è un **monoide** (non commutativo).



# Osservazione

- In base alla definizione di prodotto, ogni parola non vuota  $\alpha = x_1x_2\dots x_n$  si può scrivere in uno ed un solo modo come prodotto di parole di lunghezza 1, cioè di elementi di  $X$ .
- Ciò si esprime dicendo che:
  - $(X^*, \cdot)$  è il monoide libero generato dall'insieme  $X$ .

# Definizioni

## ■ Prefisso, Suffisso

- Se  $\gamma \in X^*$  è della forma  $\gamma = \alpha \beta$ , ove  $\alpha, \beta \in X^*$ , allora  $\alpha$  è un *prefisso* di  $\gamma$  e  $\beta$  è un *suffisso* di  $\gamma$ .

## ■ Sottostringa

- Se  $\delta, \beta \in X^*$  e  $\delta$  è della forma  $\delta = \alpha \beta \gamma$ , ove  $\alpha, \beta \in X^*$  allora  $\beta$  è una *sottostringa* di  $\delta$ .

## ■ Esempio

- Sia  $\gamma = 00110$ . Allora:
  - $\{\lambda, 0, 00, 001, 0011, \gamma\}$  è l'insieme dei prefissi di  $\gamma$
  - $\{\lambda, 0, 10, 110, 0110, \gamma\}$  è l'insieme dei suffissi di  $\gamma$
  - $\{\lambda, 0, 1, 00, 01, 10, 11, 001, 011, 110, 0011, 0110, \gamma\}$  è l'insieme delle sottostringhe di  $\gamma$ .

# Definizioni

## ■ Potenza di una stringa

- Data una stringa  $\alpha$  su  $X$ , la *potenza  $h$ -esima* di  $\alpha$  è definita (induttivamente) come segue:

$$\alpha^h = \begin{cases} \lambda & \text{se } h = 0 \\ \alpha \alpha^{h-1} & \text{altrimenti} \end{cases}$$

con  $h = 0, 1, 2, \dots$

- La potenza  $h$ -esima di una stringa è un caso speciale di concatenamento (in quanto la si ottiene concatenando una stringa  $h$  volte con se stessa).

# Definizioni

## ■ Potenza di un alfabeto

□ Sia  $X$  un alfabeto, poniamo:

1)  $X^1 = X$

2)  $X^2 = \{x_1x_2 \mid x_1, x_2 \in X, x_1x_2 \equiv x_1 \cdot x_2\}$

3)  $X^3 = \{x_1x_2x_3 \mid x_1x_2 \in X^2, x_3 \in X, x_1x_2x_3 \equiv x_1x_2 \cdot x_3\}$

..) .....

i)  $X^i = \{x_1x_2 \dots x_{i-1}x_i \mid x_1x_2 \dots x_{i-1} \in X^{i-1}, x_i \in X, x_1x_2 \dots x_i \equiv x_1x_2 \dots x_{i-1} \cdot x_i\}$

# Definizioni

## ■ Potenza di un alfabeto

- Se  $i \geq 2$  si ha:

$$X^+ = X \cup X^2 \cup \dots \cup X^i \cup \dots = \bigcup_{i=1}^{+\infty} X^i$$

- Se  $\lambda$  è la parola vuota e prendiamo un  $w \in X^+$  tale che  $w \cdot \lambda = \lambda \cdot w = w$  si ha:

$$X^* = \{\lambda\} \cup X^+$$

- Inoltre si ha:

$$X^h = \begin{cases} \{\lambda\} & \text{se } h = 0 \\ X \cdot X^{h-1} & \text{altrimenti} \end{cases}$$

# Definizioni

## ■ Linguaggio formale

- Un *linguaggio formale*  $L$  su un alfabeto  $X$  è un sottoinsieme di  $X^*$ .

$$L \subseteq X^*$$

## ■ Esempio:

- Il linguaggio delle parentesi ben formate è un linguaggio formale in quanto, denotato con  $M$  tale linguaggio, si ha:

$$M \subset \{ (, ) \}^*$$

- I linguaggi formali possono essere di natura molto diversa l'uno dall'altro.

# Esempi di linguaggi formali

- Un linguaggio di programmazione può essere costruito a partire dall'alfabeto  $X$  dei simboli sulla tastiera.
- L'insieme, finito o infinito, dei programmi ben costruiti sintatticamente (ossia, che rispettano la sintassi) costituisce un linguaggio.
- Consideriamo l'insieme dei teoremi di una teoria matematica. I teoremi sono particolari stringhe di simboli del nostro alfabeto. L'insieme dei teoremi "ben formati" rappresenta un linguaggio. Ad esempio, la stringa " $ab=ba$ " non è un teorema della teoria dei gruppi, ma della teoria dei gruppi abeliani.

# Generazione e riconoscimento di linguaggi formali

- A noi interessano i linguaggi formali da almeno due *punti di vista*
  - Descrittivo/Generativo
  - Riconoscitivo



# Generazione e riconoscimento di linguaggi formali

- *Punto di vista* **Descrittivo/Generativo**
  - Come possiamo *generare* gli elementi di un dato linguaggio  $L$ ?  
Un linguaggio finito può essere descritto/generato per elencazione degli elementi (se il numero non è troppo grande). Un linguaggio infinito non è elencabile. Questi sono i più interessanti perché devono essere specificati necessariamente attraverso una *proprietà* che ne caratterizza gli elementi, che ne definisce l'intensione. Tale proprietà può essere vista come una regola da seguire per generare gli elementi del linguaggio. Il vero problema è trovare la(e) regola(e) generativa(e) (di produzione) di un linguaggio. È quello che accade quando si impara un linguaggio: non è possibile memorizzare tutte le frasi del linguaggio.

# Generazione di linguaggi formali

## ■ Esempio

- Non è possibile “elencare” tutti i teoremi della teoria dei gruppi, perché sono infiniti i teoremi realizzabili combinando quelli noti.
- Un libro di teoria dei gruppi non è l’elencazione dei teoremi, ma fornisce una serie di assiomi e le regole con le quali, a partire dagli assiomi, è possibile costruire tutti i teoremi della teoria dei gruppi.
- Per descrivere la regola di produzione di un linguaggio, utilizzeremo una notazione insiemistica.

# Generazione di linguaggi formali

## ■ Esempio

- Sia  $L$  il linguaggio su  $X = \{0\}$  costituito da tutte e sole le stringhe che hanno un numero pari di 0, cioè:

$$L = \{\lambda, 00, 0000, 000000, \dots\}$$

La regola di produzione di  $L$  viene espressa come segue:

$$L = \{\lambda\} \cup \left\{ w^n \mid w = 00, n = 1, 2, \dots \right\}$$

# Generazione e riconoscimento di linguaggi formali

## ■ *Punto di vista* **Riconoscitivo**

- Come possiamo *riconoscere* gli elementi di un dato linguaggio  $L$ ? Questo secondo punto di vista ha come obiettivo la costruzione di “macchine” in grado di decidere/stabilire se una stringa è un elemento di  $L$  oppure no. Si intende costruire una “macchinetta” cui dare in ingresso una particolare parola e che produce una tra due possibili risposte:

$$sì \equiv ' \in L ' \quad e \quad no \equiv ' \notin L '$$



# Riconoscimento di linguaggi formali

- Esempio
  - L'esecuzione di un programma errato sintatticamente viene inibita. Questo è indice dell'esistenza di una "macchinetta" che stabilisce se il programma appartiene o no all'insieme dei programmi sintatticamente ben costruiti.

# Generazione di linguaggi formali: esempio

- Sia dato l'alfabeto:  $X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}$

Voglio generare il linguaggio  $L$  dei numeri interi relativi. Ovviamente:  $L \subseteq X^*$

Più precisamente,  $L \subset X^*$  poiché, ad esempio,

$$1 + + - 5 \notin L$$

Non possiamo elencare gli elementi di  $L$ . Cerchiamo dunque una serie di regole mediante le quali è possibile produrre tutti e soli gli elementi di  $L$ .

Assumiamo, per semplicità, che un numero relativo sia costituito da una serie di cifre precedute da  $+$  o  $-$ .

# Generazione di linguaggi formali: esempio

- Adottiamo la BNF per descrivere le produzioni:

$$\langle S \rangle ::= + \langle I \rangle \mid - \langle I \rangle$$

$$\langle I \rangle ::= \langle D \rangle \mid \langle I \rangle \langle D \rangle$$

$$\langle D \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

- Queste regole generano tutti gli interi relativi purché partiamo dal simbolo nonterminale  $S$ .
- $I$  è il simbolo nonterminale (da ora in poi, talvolta abbreviato in  $NT$ ), anche detto *categoria sintattica*, che sta ad indicare (e da cui si genera) la classe dei numeri interi.
- $I$  è definito ricorsivamente o come una cifra oppure come un intero seguito da una cifra.
- Ogni intero relativo è generato da queste regole e niente che non sia un intero relativo può essere generato da queste regole.

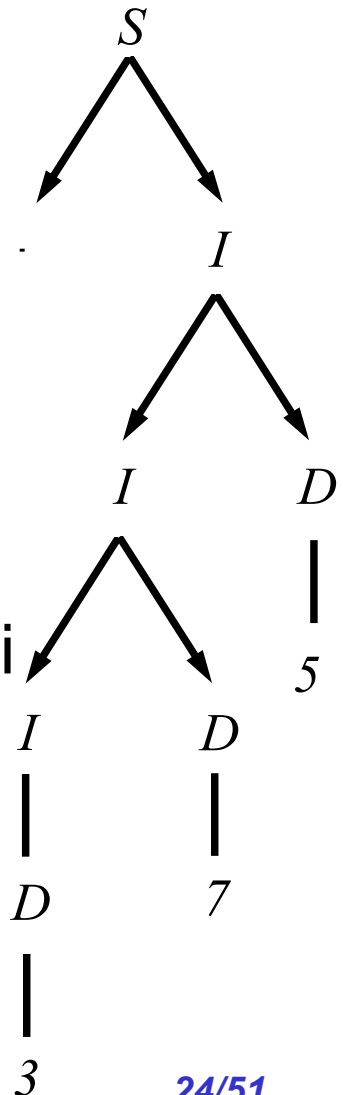
# Generazione di linguaggi formali: esempio

- Generazione ad albero:
  - proviamo a generare l'intero relativo -375
- Tale albero prende il nome di *albero di derivazione*.
- $S \Rightarrow -375 \Leftrightarrow -375 \in L$
- Nella notazione vista per il linguaggio delle parentesi ben formate, tipica per i linguaggi formali, la grammatica diventa:

$$S \rightarrow +I \mid -I$$

$$I \rightarrow D \mid ID$$

$$D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$





# Grammatiche generative

- Dagli esempi di linguaggi visti, possiamo trarre le seguenti conclusioni. Per generare un linguaggio sono necessari:
  - un insieme  $X$  di simboli primitivi con cui si formano le parole del linguaggio, detto *alfabeto dei simboli terminali* o *alfabeto terminale*;
  - un insieme  $V$  di simboli ausiliari o variabili con cui si identificano le categorie sintattiche del linguaggio, detto *alfabeto dei simboli nonterminali (ausiliari)* o *alfabeto nonterminale* o *alfabeto delle variabili*;
  - un simbolo speciale  $S$ , scelto tra i nonterminali, da cui far partire la generazione delle parole del linguaggio. Tale simbolo è detto *assioma* o *scopo* o *simbolo distintivo* o *simbolo di partenza* o *simbolo iniziale*;
  - un insieme  $P$  di **produzioni**, espresse in un formalismo quali regole di riscrittura, BNF ( $a ::= b$ ), carte sintattiche, ...

# Definizione di Grammatica generativa o a struttura di frase

- Una *grammatica generativa* o *a struttura di frase*  $G$  è una quadrupla

$$G = (X, V, S, P)$$

ove:

- $X$  è l'*alfabeto terminale* per la grammatica;
- $V$  è l'*alfabeto nonterminale* o delle variabili per la grammatica;
- $S$  è il *simbolo di partenza* per la grammatica;
- $P$  è l'insieme delle *produzioni* della grammatica ed inoltre valgono le seguenti condizioni:

$$X \cap V = \emptyset \quad \text{e} \quad S \in V$$

# Definizione di Produzione

- Una *produzione* è una coppia  $(v, w)$ ,  
ove  $v \in (X \cup V)^+$  e  $v$  contiene un  $NT \Leftrightarrow v \in (X \cup V)^* V (X \cup V)^*$   
 $w \in (X \cup V)^*$  ( $w$  può essere anche  $\lambda$ ).

*Un elemento  $(v, w)$  di  $P$  viene comunemente scritto nella forma:*

$$v \rightarrow w$$

Una produzione deve, in qualche modo, riscrivere un  $NT$ .

# Definizione di Produzione

- Per convenzione, gli elementi di  $X$  sono rappresentati di solito con lettere minuscole (con o senza pedici e di solito sono le prime lettere dell'alfabeto) o cifre ed operatori (connettivi), mentre gli elementi di  $V$  sono rappresentati con lettere maiuscole (con o senza pedici) o con stringhe delimitate dalle parentesi angolari "<" e ">".
- La notazione  $\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_k$  è impiegata come abbreviazione della seguente:

$$\alpha \rightarrow \beta_1$$

$$\alpha \rightarrow \beta_2$$

...

$$\alpha \rightarrow \beta_k$$

## Esempi di grammatiche

- La grammatica per il linguaggio delle parentesi ben formate

$$G_1 = (\{(, )\}, \{S\}, S, \{S \rightarrow ( ), S \rightarrow (S), S \rightarrow SS\})$$

- La grammatica per il linguaggio dei numeri interi relativi

$$G_2 = (\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}, \{S, I, D\}, S, \\ \{S \rightarrow +I, S \rightarrow -I, I \rightarrow D, I \rightarrow ID, D \rightarrow 0, D \rightarrow 1, \dots, D \rightarrow 9\})$$

## Definizione di derivazione o produzione diretta

- Sia  $G = (X, V, S, P)$  una grammatica e siano  $y$  e  $z$  due stringhe finite di simboli in  $X \cup V$  (stringhe di terminali e nonterminali) tali che:

$y = \gamma\alpha\delta$  e  $z = \gamma\beta\delta$ , ove  $y \in (X \cup V)^+$ ,  $z \in (X \cup V)^*$ ,  
 $\beta, \gamma, \delta \in (X \cup V)^*$   $\alpha \in (X \cup V)^+$  e  $\alpha$  contiene un NT

1) Scriviamo

$$y \Rightarrow z$$

e diciamo che  $y$  *produce direttamente*  $z$  o che  $z$  è *derivata direttamente* da  $y$  se:

$$\alpha \rightarrow \beta \in P$$

ossia se esiste in  $G$  una produzione  $\alpha \rightarrow \beta$

## Definizione di derivazione o produzione diretta

2) Scriviamo

$$y \stackrel{*}{\Rightarrow} z$$

e diciamo che  $y$  *produce*  $z$  o che  $z$  è *derivabile* da  $y$  se  $y = z$  o esiste una sequenza di stringhe

$w_1, w_2, \dots, w_n$ , con  $w_1, w_2, \dots, w_{n-1} \in (X \cup V)^+$ ,  $w_n \in (X \cup V)^*$   
 $w_1 = y$  e  $w_n = z$  tali che  $\forall i, i = 1, 2, \dots, n-1: w_i \underset{G}{\Rightarrow} w_{i+1}$   
( $w_i$  produce direttamente  $w_{i+1}$ ), cioè:

$$y \stackrel{*}{\Rightarrow} z \Leftrightarrow \begin{cases} y = z \\ \text{oppure} \\ w_1 = y \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \dots \Rightarrow w_{n-1} \Rightarrow w_n = z \end{cases}$$

# Osservazione

- La nozione di derivazione diretta stabilisce una relazione binaria in  $(X \cup V)^*$ .  
Date due stringhe  $y$  e  $z$ , il simbolo  $\Rightarrow$  può esserci o meno; dipende dall'esistenza di una produzione.  
Allora possiamo anche definire una composizione di relazioni:

$$y \stackrel{2}{\Rightarrow} z \stackrel{def}{\Leftrightarrow} \exists w : y \Rightarrow w \text{ e } w \Rightarrow z$$

dove 2 è il numero di trascrizioni necessarie per passare da  $y$  a  $z$  (ossia, la *lunghezza della derivazione*).



## Osservazione

- Da ciò si ha:  $\Rightarrow^* = I \cup \Rightarrow \cup \Rightarrow^2 \cup \Rightarrow^3 \cup \dots$

ove  $I$  è la relazione identica e  $\Rightarrow^n$  indica la composizione della relazione  $\Rightarrow$   $n$  volte con se stessa.

\*

$\Rightarrow$  è la **chiusura riflessiva e transitiva** della relazione di derivazione diretta;

+

$\Rightarrow$  è la **chiusura transitiva** della stessa relazione.

## Definizione di linguaggio generato da una grammatica

- Sia  $G = (X, V, S, P)$  una grammatica. Il *linguaggio generato da  $G$* , denotato con  $L(G)$ , è l'insieme delle stringhe di terminali derivabili dal simbolo di partenza  $S$ .

$$L(G) = \left\{ w \in X^* \mid S \xRightarrow[G]{*} w \right\}$$

- Sono, dunque, stringhe di  $L(G)$  le stringhe che:
  - consistono di soli terminali;
  - possono essere derivate da  $S$  in  $G$ .

## Definizione di forma di frase

- Sia  $G = (X, V, S, P)$  una grammatica. Una stringa  $w$ ,  $w \in (X \cup V)^*$ , è una **forma di frase** di  $G$  se:

$$S \stackrel{*}{\Rightarrow}_G w$$

- Alle forme di frase si applicano le stesse definizioni (es.: potenza) e gli stessi operatori (es.: concatenazione) dati per le stringhe.
- **Proposizione:**
  - Data una grammatica  $G = (X, V, S, P)$ ,  $L(G)$  è l'insieme delle forme di frase terminali (o *frasi*) di  $G$ .

## Definizione di grammatiche equivalenti

- Due grammatiche  $G$  e  $G'$  si dicono *equivalenti* se generano lo stesso linguaggio, ossia se

$$L(G) = L(G')$$

## Esempio

- Sia  $G = (X, V, S, P)$ , ove

$$X = \{a, b\}, \quad V = \{S\}, \quad P = \left\{ S \xrightarrow{(1)} aSb, S \xrightarrow{(2)} ab \right\}$$

Determiniamo  $L(G)$ .

$ab \in L(G)$  poiché  $S \xrightarrow{(2)} ab$

Se numeriamo le produzioni, possiamo indicare la produzione usata immediatamente al di sotto del simbolo  $\Rightarrow$ .

$\Rightarrow \equiv$  ho applicato la produzione  $n$

$\xrightarrow{(n)}$   
 $\xrightarrow{k}$

$y \Rightarrow z \equiv$   $y$  produce  $z$  in  $k$  passi, dove  $k$ =lunghezza della derivazione

## Esempio

- $a^2b^2 \in L(G)$  poiché  $S \xRightarrow{(1)} aSb \xRightarrow{(2)} a^2b^2$
- $a^3b^3 \in L(G)$  poiché  $S \xRightarrow{3} a^3b^3$

.....

$$\{a^n b^n \mid n > 0\} \subseteq L(G)$$

- Inoltre, qualsiasi derivazione da  $S$  in  $G$  produce frasi del tipo  $a^n b^n$ .
  - Dunque  $L(G) \subseteq \{a^n b^n \mid n > 0\}$  e quindi

$$L(G) = \{a^n b^n \mid n > 0\}$$

## Notazione

- Per rendere più concisa la descrizione di una grammatica, spesso ci limiteremo ad elencarne le produzioni, quando sia chiaro quale sia il simbolo di partenza e quali siano i terminali ed i nonterminali.
- Inoltre, le produzioni con la stessa parte sinistra vengono accorpate attraverso l'uso del simbolo “ | “ (preso a prestito dalla BNF).
- Infine, ometteremo l'indicazione della grammatica dalla simbologia di derivazione e derivazione diretta quando sia chiaro dal contesto a quale grammatica si fa riferimento.

## Esempio

- Sia data la seguente grammatica:

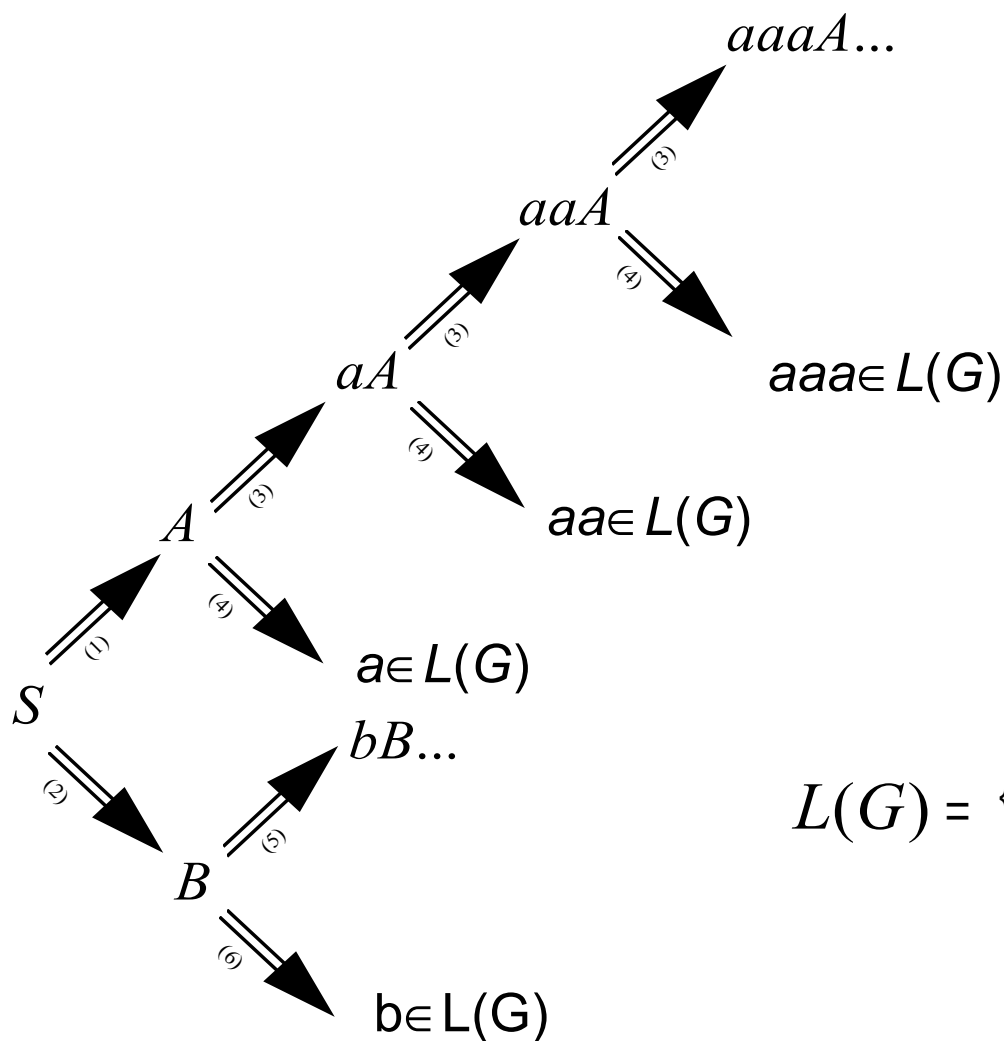
$$S \rightarrow \overset{(1)}{A} | \overset{(2)}{B}, \quad A \rightarrow \overset{(3)}{aA} | \overset{(4)}{a}, \quad B \rightarrow \overset{(5)}{bB} | \overset{(6)}{b}$$

Determinare  $L(G)$ .

Non sappiamo se applicare  $S \xrightarrow{(1)} A$  oppure  $S \xrightarrow{(2)} B$  inizialmente. I meccanismi di costruzione di un linguaggio sono generalmente **non deterministici**, poiché può non essere univoca la sostituzione da operare ad una forma di frase se uno stesso NT si trova a sinistra di 2 o più produzioni, come illustrato nella figura seguente.



# Esempio



$$L(G) = \{a^n \mid n > 0\} \cup \{b^n \mid n > 0\}$$

## Osservazione

- Dunque, una grammatica è uno *strumento generativo* di un linguaggio perché, data una qualsiasi parola di quel linguaggio, possiamo risalire mediante le produzioni al simbolo di partenza della grammatica.
- Viceversa, dato il simbolo di partenza di una grammatica, seguendo uno qualsiasi dei cammini dell'albero di derivazione, si produce una parola “valida” del linguaggio.

## Osservazione

- In generale, dato un linguaggio  $L$  ed una grammatica  $G$ , **non** esiste un algoritmo in grado di dimostrare che la grammatica genera il linguaggio, ossia che  $L = L(G)$ .  
Più specificamente, **non** esiste un algoritmo che stabilisce se una data stringa è generata o no dalla grammatica presa in considerazione.
- Tutto ciò si riassume nella seguente **proposizione**:
  - Il problema di dimostrare la correttezza di una grammatica non è risolubile algebricamente, in generale.

## Osservazione

- In molti casi importanti, però, è possibile dimostrare per induzione che una particolare grammatica genera proprio un particolare linguaggio.
- Queste dimostrazioni ci consentono di stabilire se, data una grammatica  $G$  ed un linguaggio  $L$ , risulta:
  - $w \in L(G) \Rightarrow w \in L$     cioè  $L(G) \subseteq L$
  - $w \in L \Rightarrow w \in L(G)$     cioè  $L \subseteq L(G)$

## Osservazione

- In molti casi importanti, però, è possibile dimostrare per induzione che una particolare grammatica genera proprio un particolare linguaggio.
- Queste dimostrazioni ci consentono di stabilire se, data una grammatica  $G$  ed un linguaggio  $L$ , risulta:

□  $w \in L(G) \Rightarrow w \in L$     cioè  $L(G) \subseteq L$

□  $w \in L \Rightarrow w \in L(G)$     cioè  $L \subseteq L(G)$

La grammatica  $G$  genera solo stringhe appartenenti al linguaggio  $L$ .

## Osservazione

- In molti casi importanti, però, è possibile dimostrare per induzione che una particolare grammatica genera proprio un particolare linguaggio.
- Queste dimostrazioni ci consentono di stabilire se, data una grammatica  $G$  ed un linguaggio  $L$ , risulta:
  - $w \in L(G) \Rightarrow w \in L$  cioè  $L(G) \subseteq L$
  - $w \in L \Rightarrow w \in L(G)$  cioè  $L \subseteq L(G)$

Il linguaggio  $L$  comprende solo parole generabili dalla grammatica  $G$ .

## Principio di induzione

- Sia  $n_0$  un intero e sia  $P=P(n)$  un enunciato che ha senso per ogni intero maggiore o uguale ad  $n_0$ . Se:
  - $P(n_0)$  è vero
  - Per ogni  $n > n_0$ ,  $P(n-1)$  vero implica  $P(n)$  veroallora  $P(n)$  è vero per tutti gli  $n$  maggiori o uguali ad  $n_0$

## Esercizi

- Determinare una grammatica che genera il seguente linguaggio:

$$L = \{a^n b^n \mid n > 0\}$$

e dimostrare questo risultato.

- Che tipo di grammatica genera  $L$  ?

Soluzione esercizio



## Esercizi

- Determinare una grammatica che genera il seguente linguaggio:

$$L = \{a^n b^{2n} \mid n > 0\}$$

e dimostrare questo risultato.

- Di che tipo è la grammatica che genera  $L$  ?

Soluzione esercizio

## Esercizi

- Sia data la seguente grammatica:

$$G = (X, V, S, P)$$

$$X = \{0, 1\} \quad V = \{S, A, B\}$$

$$P = \left\{ S \rightarrow \overset{(1)}{0}B \mid \overset{(2)}{1}A, \quad A \rightarrow \overset{(3)}{0} \mid \overset{(4)}{0}S \mid \overset{(5)}{1}AA, \quad B \rightarrow \overset{(6)}{1} \mid \overset{(7)}{1}S \mid \overset{(8)}{0}BB \right\}$$

- Determinare il linguaggio generato da  $G$ .

Soluzione esercizio

## Esercizi

- Dimostrare per induzione che il linguaggio  $L$  generato dalla seguente grammatica è vuoto:

$$G = (X, V, S, P)$$

$$X = \{a, b, c\} \quad V = \{S, A, B\}$$

$$P = \left\{ S \rightarrow \overset{(1)}{a} \overset{(2)}{B} S \mid \overset{(3)}{a} B \rightarrow \overset{(4)}{A} c \mid a, \overset{(5)}{b} A \rightarrow \overset{(6)}{S} \mid B a \right\}$$

Soluzione esercizio