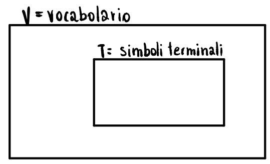
Programma: sequenze di stringhe. Stringa = sequenze di caratteri.
Abbiamo dato una definizione formale di **grammatica generativa.** Vediamone una definizione informale:

Una grammatica generativa è un insieme di regole che “specificano” o “generano” in modo ricorsivo le formule ben formate di un linguaggio. Una formula ben formata è una stringa di simboli che, intuitivamente, rappresenta un’espressione sintatticamente corretta, e viene definita mediante le regole della grammatica.

Formalmente, una grammatica è una tupla quadrupla:

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



V è il vocabolario, che comprende simboli non-terminali e simboli terminali.
T è l’insieme dei simboli terminali, sottoinsieme di V.
S è il simbolo iniziale, scelto tra i simboli non-terminali.
P è l’insieme delle **produzioni**, che indicano una possibile forma di un costrutto.

La forma delle produzioni è del tipo $\underbrace{\alpha}_{\text{DRIVER}} \to \underbrace{\beta}_{\text{BODY}}$

$\alpha \in V^+$, cioè è una stringa costituita da uno o più simboli di V, ed vincolo che almeno un simbolo $\in V \setminus T$.

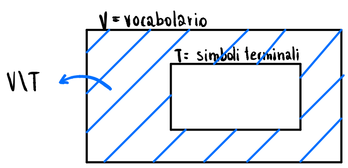
$\beta \in V^*$, ossia è una stringa costituita da 0 (stringa vuota $\beta = \varepsilon$) o più elementi.

Notazione:
- V^+: il + indica la ripetizione di una o più volte dell’oggetto alla base, la V. Si forma quindi una stringa lunga 1, 2, 3... n.
- V^* : la * indica la ripetizione di *zero* o più volte dell’oggetto alla base, la V. Si forma quindi una stringa lunga 0, 1, 2, 3... n. Quando è lunga 0, β è ε.

Un esempio di grammatica: G = ({`S, a, b`}, {`a, b`}, `S`, {`S->a, S->b`})

vocabolario di 3 simboli: nelle produzioni che vado a scrivere possono comparire solo questi 3 simboli.

Il non-terminale S viene scritto nel terminale 'a' o 'b'.



Produzione: **stringa —> stringa.**

Pesca simboli $\in V$; potrebbe essere anche ε.
E' obbligatorio che contenga un simbolo $\in V \setminus T$.

Due convenzioni:
1. I simboli non-terminali si indicano con CAPITAL letters, i simboli terminali si indicano con small letters.
2. La prima produzione ha come driver lo start symbol della grammatica.

Formalismo che indica come si deriva una qualche *parola* dalla grammatica:

data $\mu \in V^*$, μ deriva direttamente da γ in G $\Leftrightarrow \gamma = \sigma \alpha \tau$, $\alpha \to \beta \in P$, $\mu = \sigma \beta \tau$ con $\sigma, \tau \in V^*$.

Esempio, linguaggio con 2 produzioni

$$\begin{cases} S \to aSb \\ \underbrace{S}_{\alpha} \to \underbrace{ab}_{\beta} \end{cases}$$

$\gamma = \underbrace{aaa}_{\sigma}\underbrace{S}_{\alpha}\underbrace{bb}_{\tau}$ ... cosa posso derivare? Derivazione diretta: $\mu = \underbrace{aaa}_{\sigma}\underbrace{ab}_{\beta}\underbrace{bbb}_{\tau}$

Altro esempio: $\gamma = \underbrace{}_{\sigma}^{\text{vuota}}\underbrace{S}_{\alpha}\underbrace{bb}_{\tau}$, $\mu = \underbrace{}_{\sigma}\underbrace{ab}_{\beta}\underbrace{bbb}_{\tau}$

Il linguaggio di una grammatica è generato operando un passo di riscrittura dietro l’altro, fino a che nulla è più riscrivibile, ossia non esiste più alcuna stringa contente simboli non-terminali che possa essere riscritta in una stringa di simboli terminali mediante una riduzione.

La stringa μ deriva da γ in G se esiste una sequenza di stringhe $\alpha_0, \dots, \alpha_n$ tali che $\gamma = \alpha_0$, $\mu = \alpha_n$, α_{i+1} deriva direttamente da α_i ($\forall_i \; 0 \le i \le n-1$).

Ossia, $\begin{cases} S \to aSb \\ S \to ab \end{cases}$ Supponiamo di partire con $\gamma = aaSbb$. - Utilizzando la prima produzione, deriviamo $\mu = aaaSbbb$ (μ deriva direttamente da γ).
- Utilizzando la seconda: $\mu' = aaa\,ab\,bbb$ (derivazione da γ non diretta).

Il linguaggio generato dalla grammatica G = (V, T, S, P) è definito come $L(G) = \{w \mid w \in T^* \text{ e } \underbrace{S \overset{+}{\Rightarrow} w}_{\text{Da S derivo in 1 o più passi}}\}$

parole

$L(G) = \{ w \mid w \in T^* \text{ e } S \overset{+}{\Rightarrow} w \}$

Se è lunga 0, è la ε. Stella: ammette questo caso

Fosse stato *, ci sarebbe stata derivazione $S \overset{0}{\Rightarrow} S$, che non serve a derivare parole del linguaggio.

Esempi vari

$G_1: \; S \to a \qquad L(G_1) = \{a\}$

$G_2: \; S \to a, \; S \to b \qquad L(G_2) = \{a, b\}$

$G_3: \; S \to B, \; S \to a \qquad L(G_3) = \{a\}$ La B la posso riscrivere, ma non posso andare più avanti

$G_4: \; S \to B \qquad L(G_4) = \emptyset$

$G_5: \; S \to \varepsilon \qquad L(G_5) = \{\varepsilon\}$

La parola vuota è una parola? Sì.

Il linguaggio è un insieme che contiene la parola vuota, ma che non è vuoto.

$G_6: \; S \to aSb, \; S \to ab \qquad L(G_6) = \{a^n b^n \mid n > 0\}$

$S \to ab$
$S \to aSb \to a\,ab\,b$
$aSb \to aaSbb \to aa\,aSb\,bb$
$aaSbb \to \dots$

$G_7: \; S \to aSb, \; S \to \varepsilon \qquad L(G_7) = \{a^n \cancel{x}\, b^n \mid n \ge 0\}$

SUPERFLUO!

Ricordiamo che: 1) $ab = \varepsilon ab, \; \varepsilon a \varepsilon b, \dots$ sono OMONIMI: "depuriamoli" da ε!

2) Elemento neutro moltiplicazione: $\cdot$

Elemento neutro concatenazione: ε

G_8:

$S \to aaSb \qquad S \overset{k}{\Rightarrow} a^{2k} S b^k \quad k \ge 0$

$S \to A \qquad S \overset{k}{\Rightarrow} a^{2k} A b^k$

$A \to aAbb \qquad A \overset{k,j}{\Rightarrow} a^{2k} a^j A b^{2j} b^k \quad j \ge 0$

$A \to c \qquad A \overset{k,j}{\Rightarrow} a^{2k} a^j c\, b^{2j} b^k$

$L(G_8) = \{ a^{2k+j} c\, b^{2j+k} \mid j, k \ge 0 \}$

Finora abbiamo visto grammatiche libere.

Eccone una non libera:

G_9:

$S \to aAb \qquad L(G_9) = \{a^n b^n \mid n > 0\}$

$aA \to aaAb$

$A \to \varepsilon$