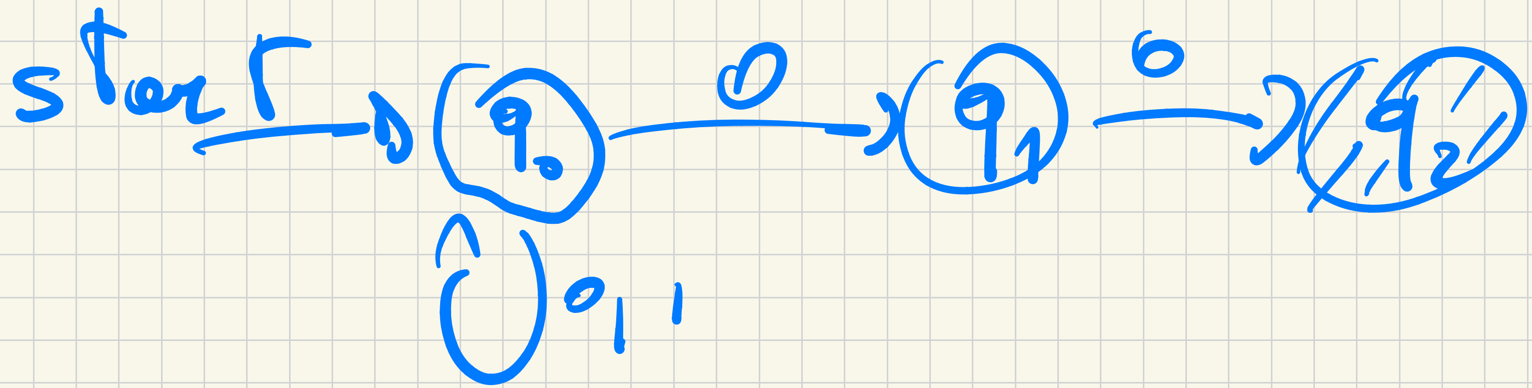
ESERCIZI

*) Progettare un automa per

$$L = \{ w \in \{0,1\}^* : w \text{ termina con '00'} \}$$



M

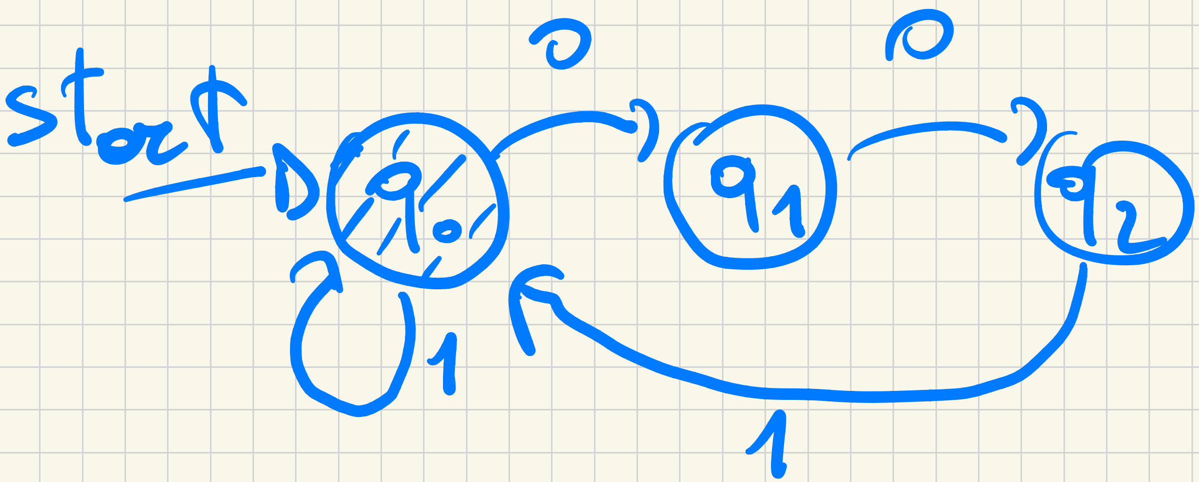
Correttezza: $w \in L$ sse M accetta w.

($\Rightarrow$) Se $w \in L$, allora M accetta w.
Per induzione: $w = 00$ è il caso base
e chiaramente M accetta 00.
Passo induttivo: $w = w'00$, $w' \in \{0,1\}^*$.
Anche in questo caso è chiaro che M
accetta w.
($\Leftarrow$) Se M accetta w, allora $w \in L$.
Per ipotesi, se M accetta w $\exists$ ramo
di computazione che termina in q_2.
Ovvero w termina con '00' e $w \in L$.

*) Consideriamo $r = 1^*(001^+)^*$

(1^+ come 1^* ma $\geq$ una occorrenza).

Progettare automa per $L(r)$ con 3 stati.



$w = 000$

q_1 q_2 REJ

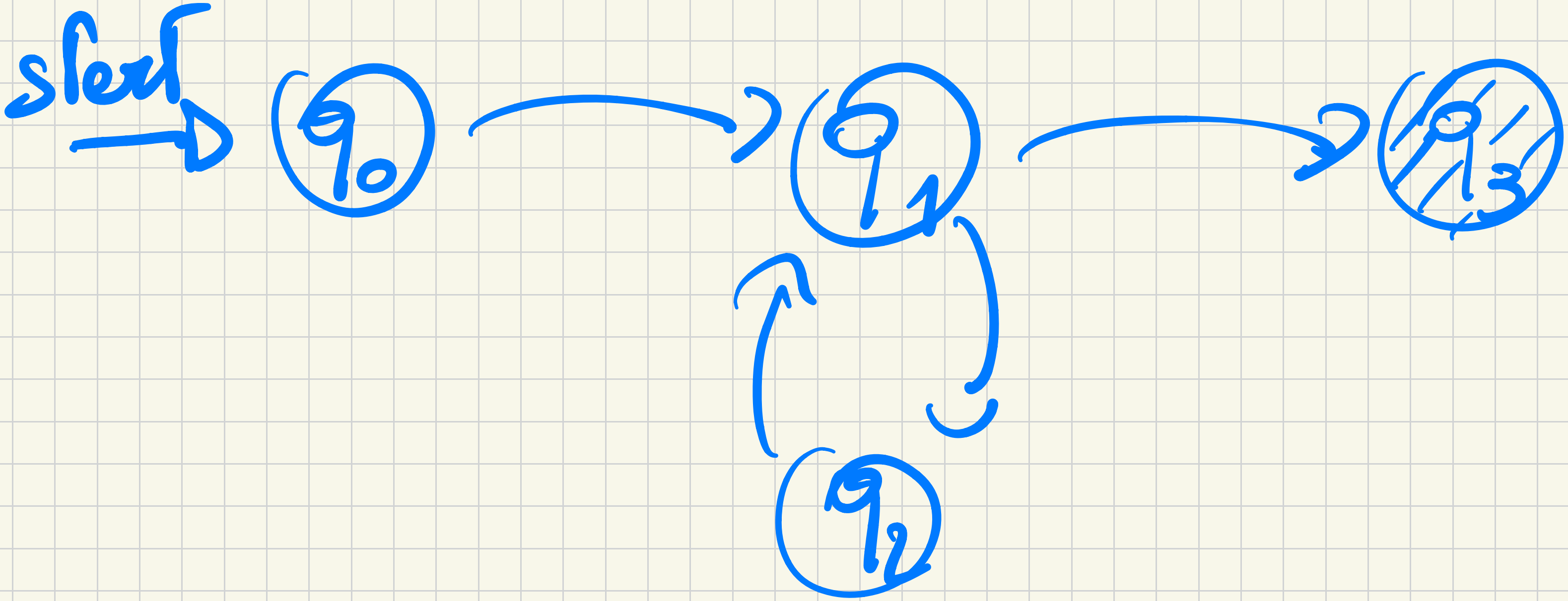
Se non c'era il 1^+, potrebbero servire 4 stati. Infatti $0011 \notin L(r')$ dove $r' = 1^*(001)^*$

*) Data $w = w_1, \dots, w_n$ su Σ sia $w^R = w_n w_{n-1} \dots w_1$. Sia L regolare.
Mostrare che $L^R = \{ w^R : w \in L \}$ è regolare.

Idea? Scambiare stato iniziale e finale (wlog ce ne è uno solo) e invertire gli archi.

Per casa: Da formalizzare.

Ad es:

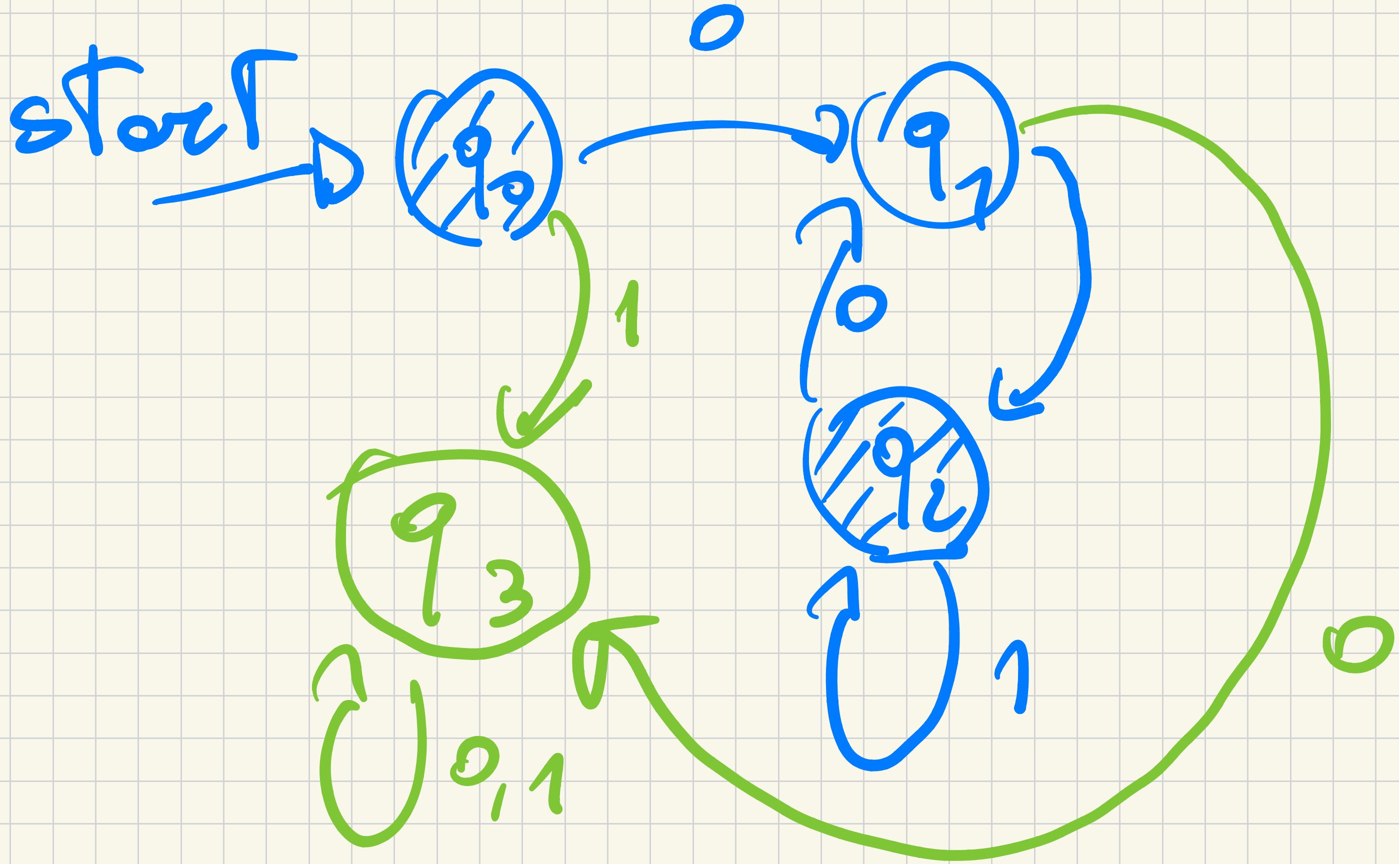


*) Costruire DFA per

$$L = \{ w \in \{0,1\}^* : w \notin (01^+)^* \}$$

Uso le chiusure per il complemento.

Prima faccio DFA per $(01^+)^*$:



Se facessi complemento di questo DFA non otterrei la negazione di $(01^*)^*$

(Per invertire stato accettante e non.)

Importante: Per applicare la chiusura è bene che il DFA sia definito su tutto gli stessi.

*) $L = \{ w w^R : w \in \{0,1\}^* \}$ non è REGOLARE.

Uso il PUMPING LEMMA. Se lo fosse $\exists\, p$ t.c. valgono le condizioni del lemma.

Considero $w = \underbrace{0^p 1}\,\underbrace{1 0^p} \in L$. Prendo

$|w| \geq p$

scomposizione $w = xyz$ t.c. $|y| > 0$

$|xy| \leq p$

$\Rightarrow$ y è fatto di solo 0.

$$w = 0^k 0^l 0^m 1 1 0^p$$

$$k + l + m = p \; ; \quad x = 0^k$$

$$k \geq 0 \qquad y = 0^l$$

$$l > 0 \qquad z = 0^m 1 1 0^p$$

Per $i = 2$, $xy^2z =$

$$= 0^k 0^{2l} 0^m 1 1 0^p \quad \text{con}$$

$$k + 2l + m = k + l + m + l$$

$$= p + l > p \rightarrow\leftarrow$$

*) $L = \{ 1^{n^2} : n \geq 0 \}$ non è REGOLARE.

Suppongo lo sia e scelgo $w = 1^{p^2} \in L$

Considero una scomposizione generica

$w = xyz$ t.c. $|y| > 0$ e $|xy| \leq p$.

In particolare:

$$w = 1^k 1^l 1^{p^2-k-l}$$

$$x = 1^k$$

$$y = 1^l$$

$$z = 1^{p^2-k-l}$$

$$k \geq 0 \ ; \ l > 0 \ ;$$

$$k + l \leq p$$

Pumping con $i=2$.

$$xy^2z = 1^k 1^{2l} 1^{p^2-k-l}$$

$$|xy^2z| = k+2l+p^2-k-l$$

$$= p^2+l > p^2$$

Però:

perché

$k+l \le p$

$$(p+1)^2 = p^2+2p+1$$

$$\ge p^2+2(k+l)+1 = p^2+2k+2l+1$$

$$> p^2+l$$

$\Rightarrow \quad |xy^2z| = p^2 + \ell < (p+1)^2$

$|xy^2z| > p^2 \longrightarrow \Leftarrow$.

GRAMMATICHE ACONTESTUALI

Introdurremo un modello di computazione più potente, utile in diverse applicazioni (es. parser per i compilatori).
Vedremo che le grammatiche corrispondono anche con un diverso tipo di automa.
Ad es. sarà molto facile dare una grammatica che genera le stringhe $0^n 1^n$ con $n \geq 0$. Facciamo subito questo come esempio.

Una grammatica è una sequenza di sostituzioni e produzioni:

REGOLE DELLA GRAMMATICA

$$\begin{cases} A \to 0A1 & (R_1) \\ A \to B & (R_2) \\ B \to \# & (R_3) \end{cases}$$

$0, 1, \#$ sono terminali

A, B sono variabili.

La grammatica genera stringhe:

-) Scrivo variabile INIZIALE

-) Sostituisco variabili con altre

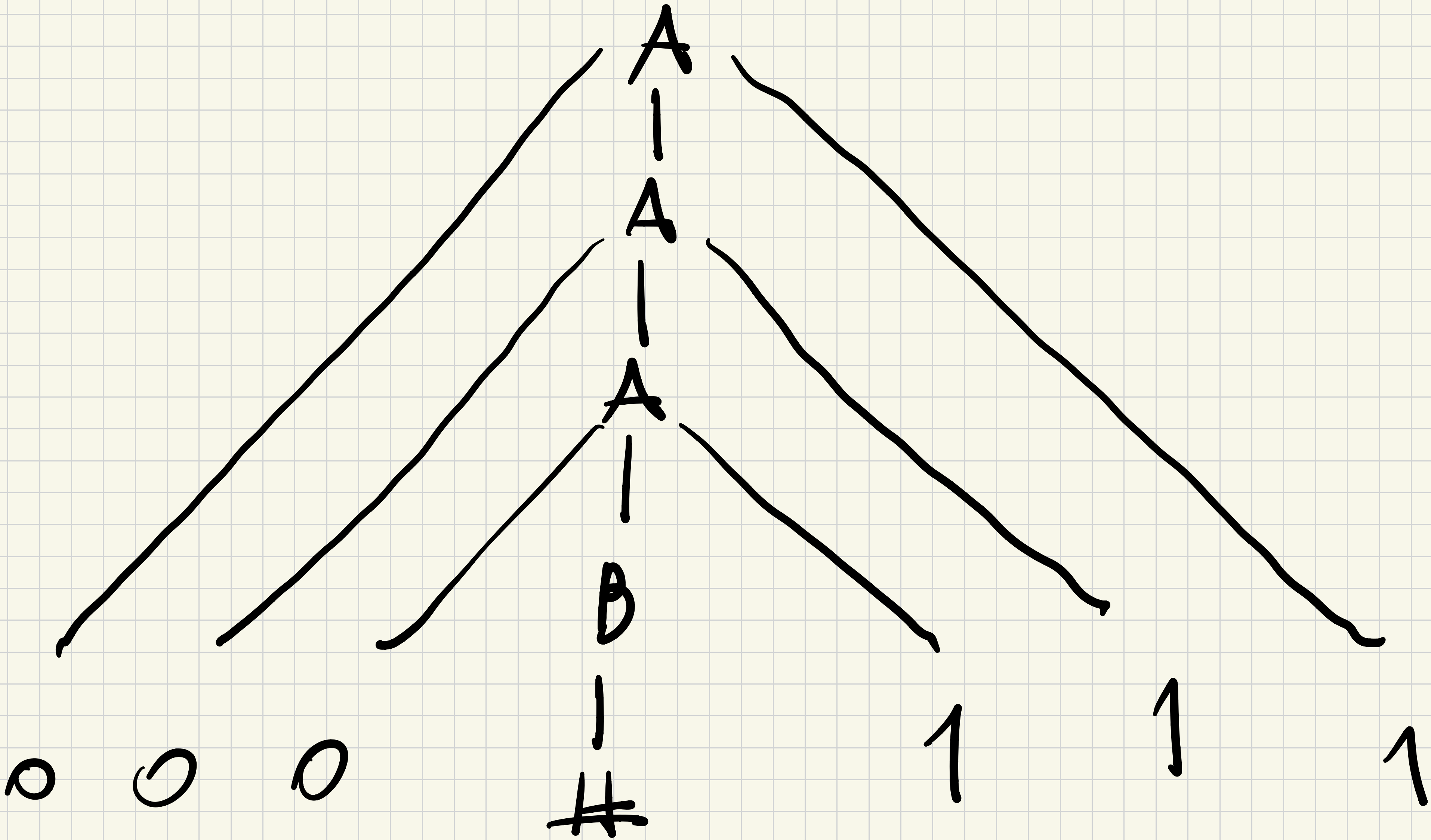
espressioni seguendo le regole della grammatica

-) Ripeto fino a che non ci sono più variabili.

Nell'esempio:

$$A \xRightarrow{R_1} 0A1 \xRightarrow{R_1} 00A11 \xRightarrow{R_1} 000A111 \xRightarrow{R_2} 000B111 \xRightarrow{R_3} 000\#111$$

Per ogni tale produzione posso associare
ALBERO SINTATTICO:



Altra grammatica G:

(R_1) $E \to E + E$

(R_2) $E \to E * E$

(R_3) $E \to (E)$

(R_4) $E \to 0 \mid 1 \mid 2 \mid \cdots \mid 9$

Terminali sono: $0, 1, \ldots, 9, +, *, (,)$

Variabili: E

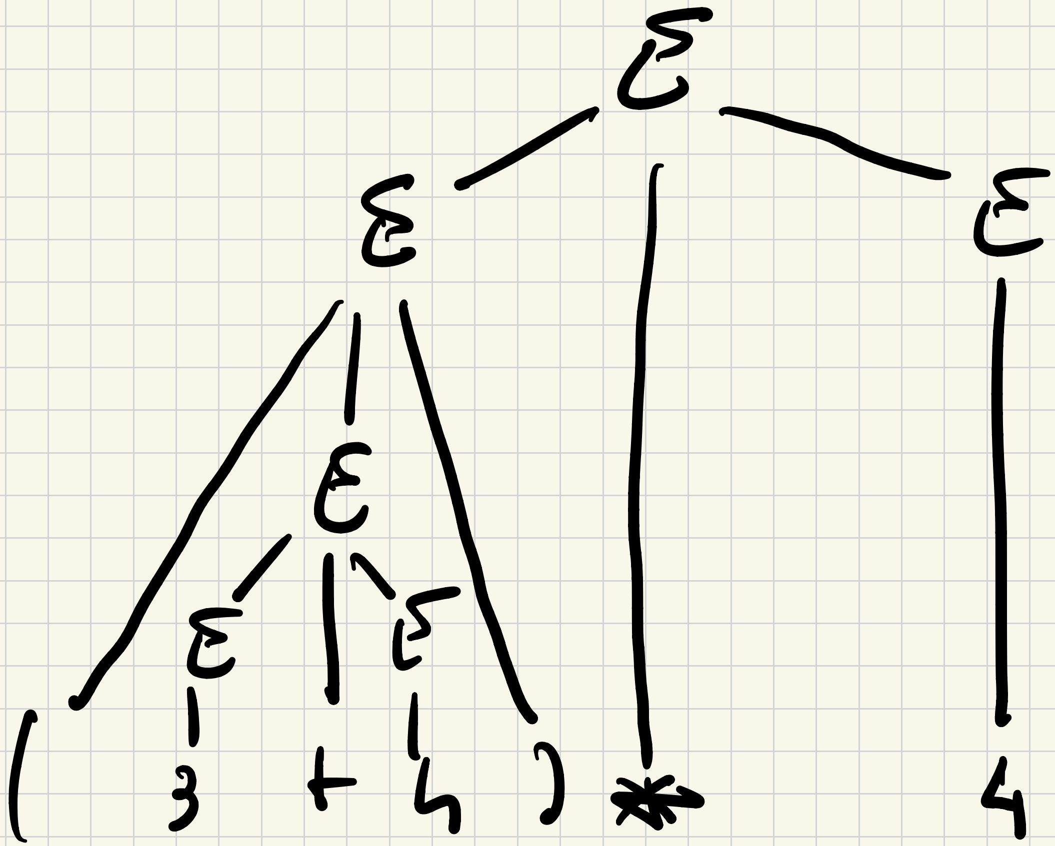
G genera $(3+4)*4$ ma non

genera $(3+4+)*4$

$$E \xRightarrow{R_2} E*E \xRightarrow{R_3} (E)*E \xRightarrow{R_1} (E+E)*E$$

$$\xRightarrow{R_4} (E+E)*4 \xRightarrow{R_4} (3+E)*4$$

$$\xRightarrow{R_4} (3+4)*4$$



DEF (CFG) Una CFG è una tripla (V, Σ, R, S) dove:

- V è insieme finito variabili
- Σ è insieme finito terminali ($V \cap \Sigma = \emptyset$)
- R è insieme finito di regole
- $S \in V$ è variabile iniziale.

Se $u, v, w \in \Sigma \cup V$ e $(A \to w) \in R$ diremo che uAv produce uwv e lo scriviamo $uAv \Rightarrow uwv$.

Diciamo che u deriva v, $u \overset{*}{\Rightarrow} v$ se:

- $u = v$, oppure
- $\exists\ u_1, \dots, u_k$ $k \geq 0$ t.c.

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \dots \Rightarrow u_k \Rightarrow v.$$

Questo ci permette di definire il linguaggio associato ad una CFG.

Sia $G = (V, \Sigma, R, S)$, allora

$$L(G) = \{ w \in \Sigma^* : S \overset{*}{\Rightarrow} w \}.$$

Esempio. $G = (V = \{S\}, \Sigma = \{a, b\}, R, S)$

$R: \quad S \to aSb \mid SS \mid \varepsilon.$

$ab \in L(G) : S \Rightarrow aSb \Rightarrow a\varepsilon b = ab$

$aabb \in L(G) : S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

Controlla che a b siano ben "annidate".

Iniziamo a vedere alcune tecniche per costruire grammatiche:

1) Unione di grammatiche

2) Da DFA alle grammatiche

3) Sfruttare ricorsione.