



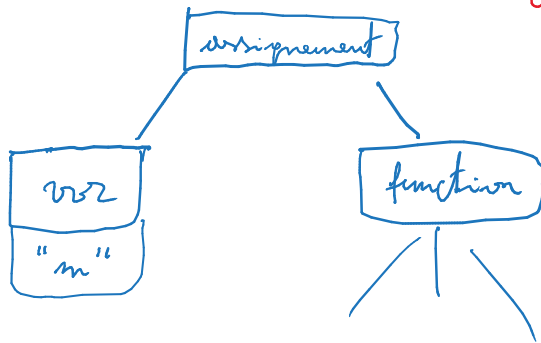
↓ non compilare

→ sequenza corretta

tokenization = analisi lessicale



analisi sintattica = parsing



esecuzione
interpretazione

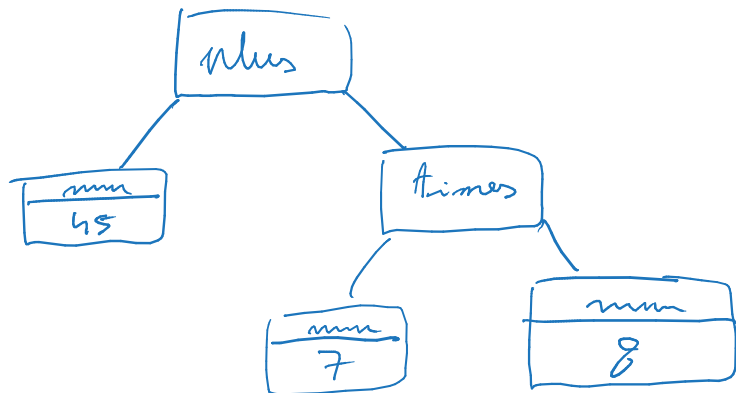
RESULTATO

$$45 + 7 * 8$$

(, tokenization

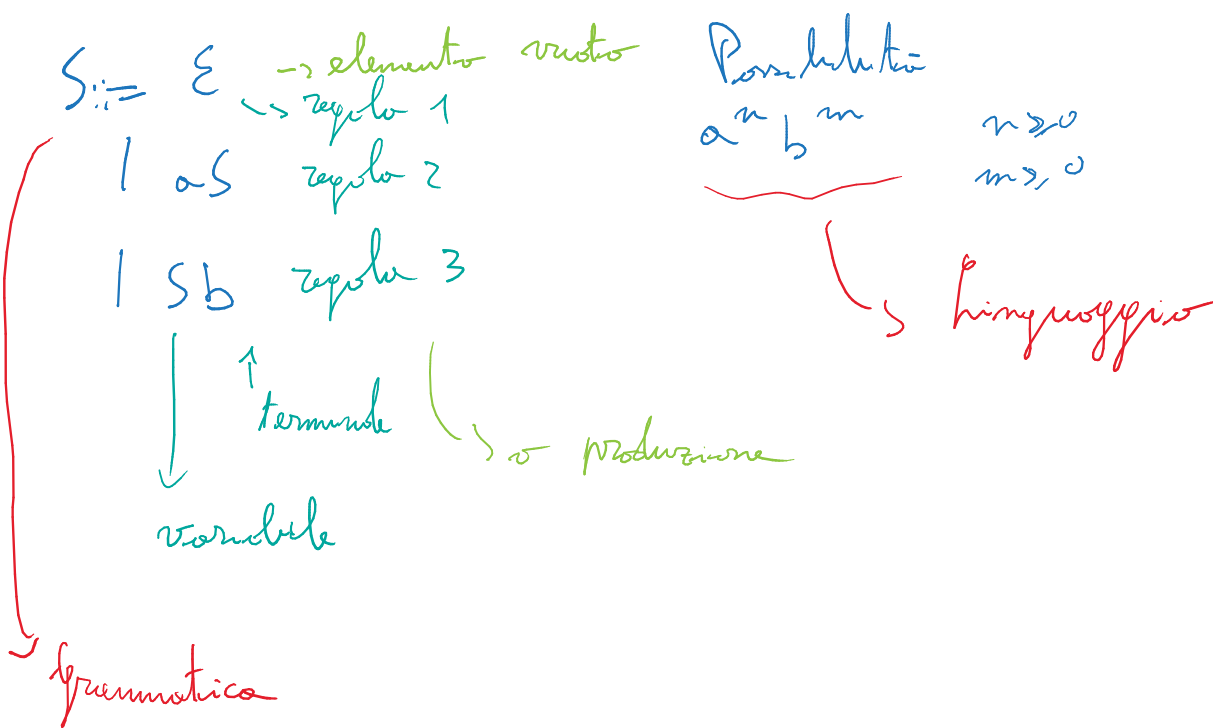


(, parsing



interpretazione

→ 101



Esercizio: scrivere grammatica per il linguaggio di parentesi

$(())$ ✓

$) ($ ✗

$()$ ✓

$() () ()$ ✓

Variante 1

$S ::= \epsilon$
 $| (S) S$

Variante 2

$S ::= \epsilon$
 $| (S)$
 $| SS$

1) Grammatiche diverse possono produrre lo stesso linguaggio

2) Per il power, una grammatica o un'altra può fare differenza
(nel caso siano presenti delle alternative, come
nella seconda variante)

$$n \geq 0$$

$$a^n b^n c^n$$

$S ::=$

$$S ::= a S B c$$

$$c B ::= B c$$

$$b B ::= b b$$

$$S \Rightarrow a \underline{S} B c \Rightarrow a a \underline{b c} B c \Rightarrow$$

$$\Rightarrow a a \underline{b B c c} \Rightarrow \underline{a a b b c c} = \text{corrisponde al linguaggio richiesto}$$

$$\{a^{2k} \mid k \geq 0\} = \{a \ a a \ a a a \ a a a a a a \dots\}$$

A, B indicatori di inizio e fine

C cursore che si muove verso destra \rightarrow

D " " " " " sinistro

S variabile di potenza

E cursore per messaggio finale

$$S ::= A a C B$$

$$C B ::= D B \quad C B ::= E$$

$$a D ::= D a \quad a E ::= E a$$

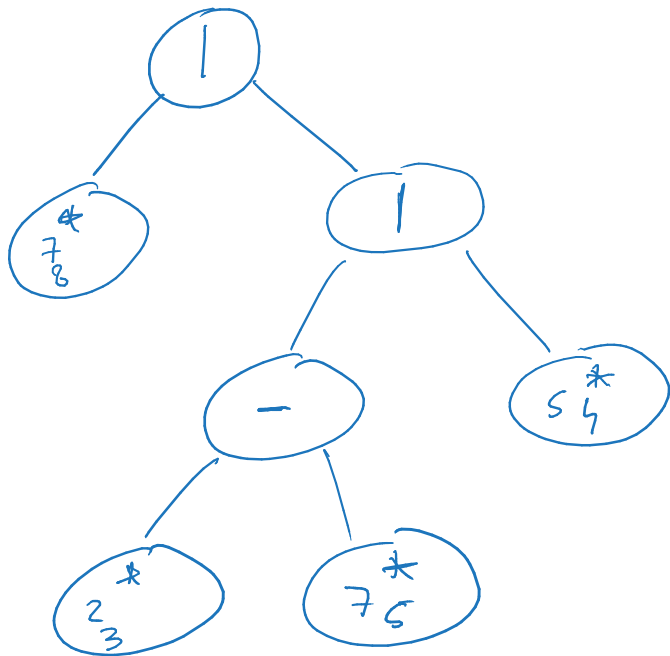
$$A D ::= A C \quad A E ::= E$$

$$C a ::= a a C$$

1) Una grammatica complessa ha lo stesso potere espressivo di un pushdown language

2) Le nostre produzioni avranno a sinistra solo variabili

Gruppi di quadrati



28/09/18

$$4 - 3 * 8 \quad (4 - 3) * 8$$

$$3 * 8 - 4 \quad 3 * (8 - 4)$$

$$E ::= E \ominus T \mid T \quad \text{— termini del linguaggio}$$

$$T ::= T \otimes F \mid F$$

$$F ::= \text{NUM} \mid (E)$$

Parser bottom-up: più potenti ma complicati da realizzare

Parser top-down: più semplici. Si parte dalla prima produzione

$$E ::= T (\ominus T)^* \rightarrow \text{ripete 0 o più volte}$$

\rightarrow solve il problema della ricorsività della prima regola inserendo la ripetitività

$$T ::= F (\otimes F)^*$$

— termini del linguaggio

$$F ::= \text{NUM} \mid (E)$$

12/10/18

Tokenizer

context-free grammars: corresponds to machines or rules

caso generale $O(n^3)$

caso normale $O(n)$

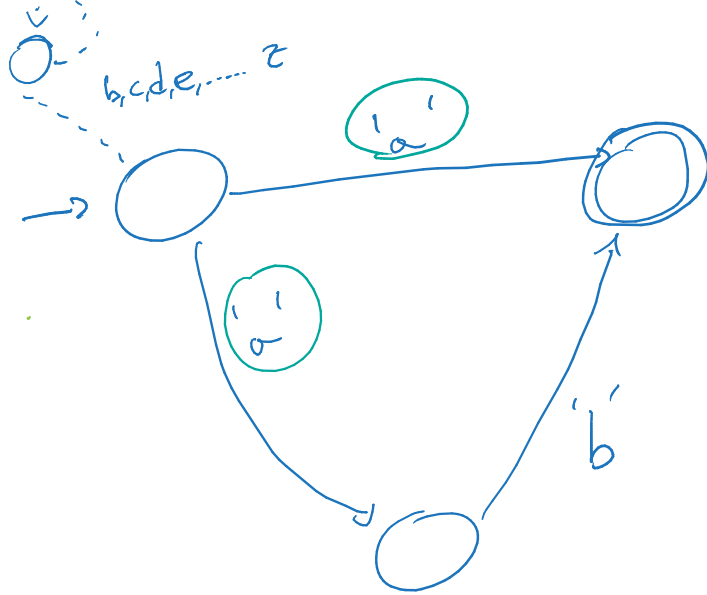
$E ::= \text{token}^*$

$E ::= \text{token}^* E$

$E ::= T \text{ token}^*$

↓ grammatiche regolari: - automa a stati finiti
- no memoria

↓ espressioni regolari



macchina non
deterministica: non so quale
percorso seguire

accetta a
 accetta ab

} esiste cammino che va allo stato
 iniziale o stato finale

Regular expression

$$RE_1 = a$$

$$RE_2 = a^* b^*$$

RE

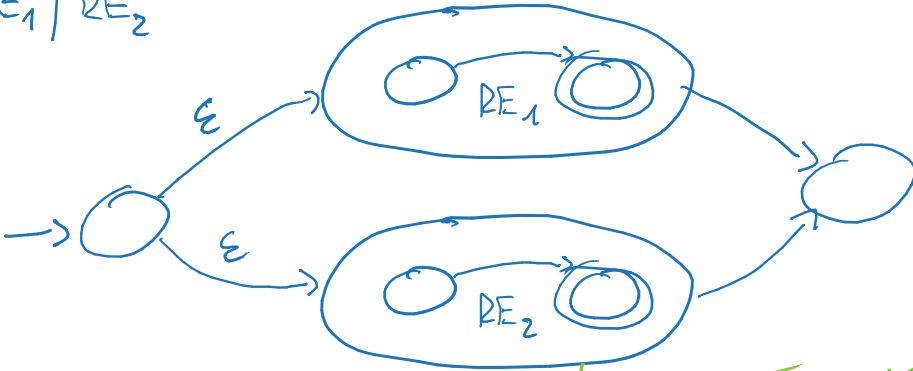
ϵ

a

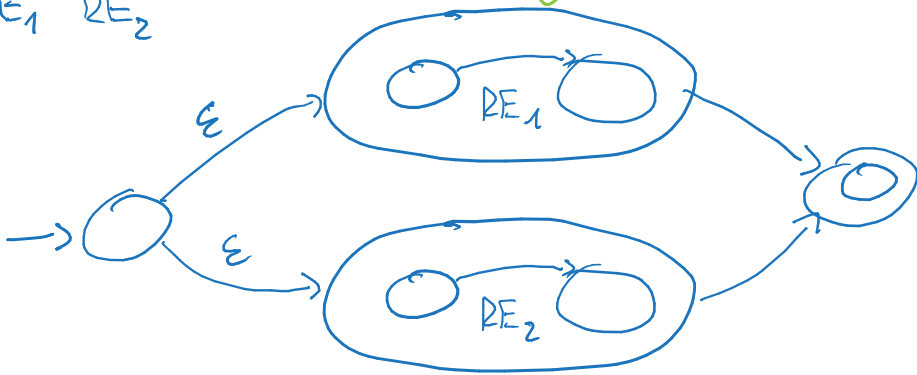
NFA : automa non deterministico



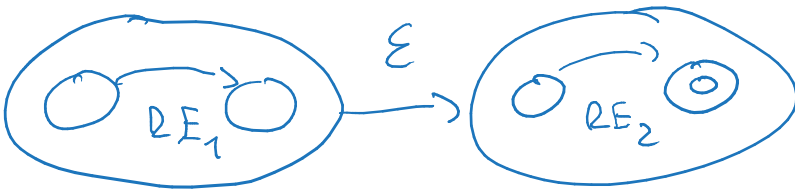
RE_1 / RE_2



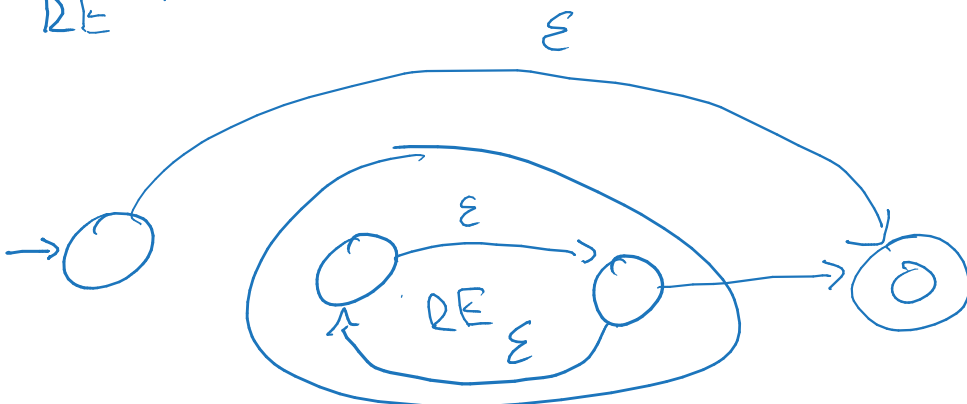
$E_1 RE_2$



$RE_1 RE_2$



RE^*



Per il power:


$e_1 || e_2 \Rightarrow$ if e_1 then true else e_2 fi

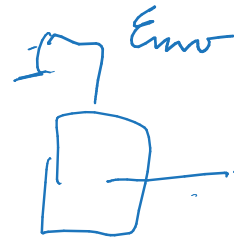
$e_1 \ll e_2 \Rightarrow$ if $!e_1$ then false else e_2 fi

$x += 5; \Rightarrow x = x + 5 \Rightarrow$ volinto solo in
alcuni linguaggi

16/11/12

Interpate

my Counter \rightarrow 



make Counter \rightarrow , my Counter \rightarrow nil

your Counter \rightarrow nil $n \rightarrow$ nil