GRAMMATICHE

la teoria delle grammatiche generative consiste in un modo oggettivo e MATEMATICO di generare stringhe, dato un dato di simboli NON TERMINALI, TERMINALI, e di associazioni.

ES si vuole generare una grammatica tale che generi le stringhe

GG/MM/AAAA.

esisterà il concetto di assioma, "punto di partenza"

data → giorno / mese / anno
giorno → num num
mese → num num
anno → num num num num

num → 0|1|2|3|4|5|6|7|8|9

le stringhe 06/09/2021 va bene, ma ad es
01/01/21 NO!

Questo serve a fare l'analisi SINTATTICA, non semantica
ES.

il cane coda NON SINTATTICAMENTE CORRETTO

cane vola cibo SINTATTICAMENTE CORRETTO
SEMANTICAMENTE ERRATO

Una grammatica è una quadrupla

$$\langle V, T, S, P \rangle$$

dove:

- V è l'insieme di simboli NON TERMINALI, non vuoto, chiamato anche VARIABILI SINTATTICHE
- T è insieme di simboli TERMINALI, non vuoto, disgiunto da V
- S è il simbolo iniziale, detto ASSIOMA
- P è un insieme finito di associazioni $\alpha \rightarrow \beta$ dove $\beta \in V \cup T$ e α contenente almeno un simbolo N.T.

Una produzione indica che ogni occorrenza destra di α può essere sostituita con la produzione di β

E' definizione RICORSIVA, poiché, dato il simbolo iniziale (solitamente caratterizzato da S), si applicano tutte le sostituzioni applicate dalle produzioni.

L'insieme di tutte le sequenze terminali $x \in T$ che sono generabili a partire da S formano il linguaggio generato dalla grammatica G, ovvero $L(G)$

ES. data la grammatica

$S \rightarrow aSb$
$S \rightarrow ab$

è facile vedere che questa corrisponde a dire che

$$\{x \text{ t.c. } x = a^n b^n, n \geq 1\} \in L(G)$$

GERARCHIA DI CHOMSKY

in base alle restrizioni che vengono imposte sul tipo di produzione che si possono ottenere le grammatiche si possono individuare in 4 CLASSI:

GRAMMATICHE REGOLARI (o di tipo 3)

sono di tipo lineare a DESTRA, quindi come primo carattere avranno SEMPRE un simbolo terminale.

$$A \to aB \quad \text{oppure} \quad A \to a$$

GRAMMATICHE LIBERE DA CONTESTO (tipo 2)

è di tipo in cui il termine a destra è libero da contesto, o restrizioni sul tipo, quindi:

$$A \to \alpha$$

GRAMMATICHE CONTESTUALI (tipo 1)

la produzione della grammatica segue la regola

$$\alpha \to \beta \quad \text{con} \quad |\beta| \geq |\alpha|$$

GRAMMATICHE NON LIMITATE (tipo 0)

sono quelle per cui NON ci sono regole e vincoli.

N.B Un linguaggio è

- REGOLARE se di tipo 3
- LIBERO da CONTESTO se di tipo 2
- CONTESTUALE se di tipo 1

LINGUAGGI DI TIPO 0 e SEMIDECIDIBILITÀ

- Per ogni linguaggio L di tipo 0 esiste una macchina di Turing NON DETERMINISTICA a due nastri che semi-decide L

  in uno c'è la stringa x, nell'altra si applica in maniera NON DETERMINISTICA ogni produzione e ad ogni iterazione si confronta con x. Se uguale termina, altrimenti CONTINUA

LINGUAGGI DI TIPO 1 e DECIDIBILITÀ

Le grammatiche di tipo 1 sono decidibili, e la dimostrazione sta nel fatto che data la prod. $\alpha \to \beta$, con $|\beta| \geq |\alpha|$, si può costruire una MdT tale per cui, per ogni input data, essa termini

- Per ogni linguaggio di tipo 1 esiste una MdT "T" a 3 nastri tale che $L(T) = L$, e per ogni input, ogni cammino di computazione T con input x TERMINA

Tipo di ling.	T.po di prod	Modello di Calcolo
tipo 0	$\alpha \to \beta$ con $\alpha \in (V \cup T)^* V (V \cup T)^*$ $\beta \in (V \cup T)^*$	MdT
Contestuale	$\alpha \to \beta$ con $\alpha \in (V \cup T)^* V (V \cup T)^*$ $\beta \in (V \cup T)^+$ $\lvert\alpha\rvert \leq \lvert\beta\rvert$	MdT lineare
priva di contesto	$A \to \beta$ con $A \in V$ e $\beta \in (V \cup T)^+$	parser
regolare	$A \to aB$ e $A \to a$ $A, B \in V$ e $a \in T$	automa a stati finiti (dopo)

LINGUAGGI REGOLARI

Uno delle principali componenti di un compilatore è ANALIZZATORE LESSICALE, o PARSER, che ha il compito di, dato un certo codice al suo interno, verificare che sia sintatticamente corretto. Per fare ciò esso identifica all'interno del codice i TOKEN, ovvero le unità significanti da analizzare.

C'è CORRISPONDENZA DIRETTA tra gli automi a STATI FINITI e le grammatiche regolari

È la prima parte di un compilatore che si occupa dell'analisi lessicale, dove il codice sorgente viene scomposto in token. Si utilizzeranno gli automi e le regex.

Secondo passo del compilatore è l'analisi SINTATTICA, dove si determina la struttura del programma, prossimo agg.

Il principale compito consiste nel trasformare sequenza di codice nei suoi relativi token. Per esempio,

```
if (x == y*(5-a)) x = 0;
```

l'analizzatore deve saper isolare
- parole chiave ("if")
- identificatori ("a, x, y, b")
- gli operatori ("==, *, -, =")
- eventuali simboli

ogni identificatore è uguale ad un altro, così come tutti gli altri pezzi di token. Spetterà più tardi, al compilatore, l'assegnazione dei giusti valori per ogni token.

Si tratti la distinzione. Il tipo generico, passato all'analizzatore sintattico, è detto TOKEN, mentre le specifiche istanze del token sono dette LESSEMI

ad esempio, si hanno 4 istanze (a, b, x, y) del token "identificatore" chiamato id.

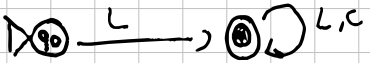
quindi, l'analizzatore LESSICALE trasforma la riga di codice in

$$if\ (id == id * (id - id))\ id = int$$

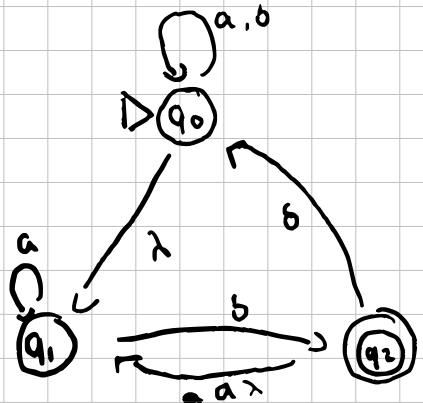
Ci sarà una tabella di simboli che assegnerà ad ogni token il relativo lessema.

l'automa a stati finiti è lo strumento più utile per la costruzione di analizzatori lessicali.

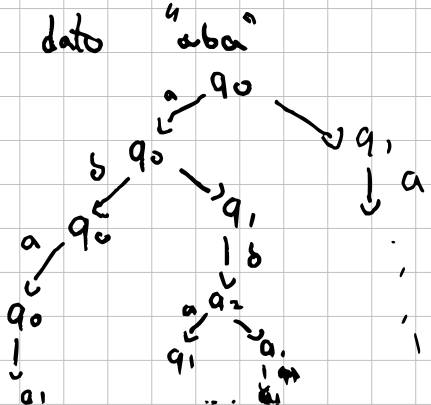
ES. L = lettera C = cifra



analizzatore token identificatore



dato "aba"



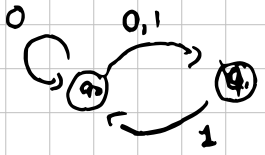
Come con le MdT, la transizione di un automa NON DETERMINISTICO può portare a più stati.

La computazione di esso CONSISTE nella costruzione di un ALBERO di computazione, dove vengono mostrati TUTTI i cammini possibili: data una stringa x_i essa è accettata se $\exists$ almeno un cammino tale che si va in uno stato FINALE

Ogni automa NON deterministico è simulabile da uno deterministico, utilizzando la tecnica della visita in ampiezza

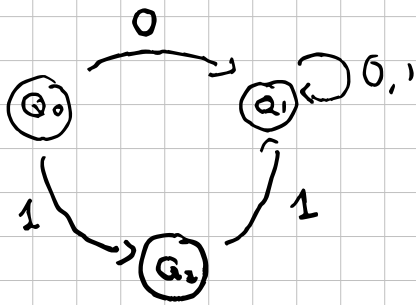
ES. dato T:

q_0	0	$\{q_0, q_1\}$
q_0	1	$\{q_1\}$
q_1	1	$\{q_0, q_1\}$



è possibile generare uno T'

Q_0	0	Q_1
Q_0	1	Q_2
Q_1	0	Q_1
Q_1	1	Q_1
Q_2	1	Q_1



0 0 0 1

Espressioni regolari: identificate come segue.

L'insieme delle espressioni regolari su un alfabeto Σ è definito induttivamente.

- Ogni carattere di Σ è un espressione regolare
- λ è un espressione regolare (carattere vuoto)
- Siano R e S espressioni regolari, allora
  - la concatenazione $R \cdot S$ è espressione regolare
  - la selezione $R+S$ è espressione regolare
  - la chiusura di Kleene ("stella") è espressione regolare
- Solamente le espressioni formate da questi sono espressioni regolari

Un espressione regolare genera un linguaggio $L(R)$

L) GRAMMATICHE REGOLARI: sono grammatiche le cui regole di produzione sono LINEARI A DESTRA. Sono equivalenti alle espressioni regolari e agli automi a stati finiti

$S \rightarrow aS$
$S \rightarrow bC$
$S \rightarrow b$
$C \rightarrow cC$
$C \rightarrow c$