Università di Firenze - Laurea Triennale in Informatica
Corso di *Laboratorio di Architettura degli Elaboratori*

A.A. 2007/2008

Progetto: Grammatiche Regolari, Automi a Stati Finiti e riconoscimento di stringhe

Grammatiche generative. Con il termine *grammatica generativa* si intende un particolare insieme di regole formali per un particolare linguaggio. Ad esempio si può parlare di una grammatica generativa dell'italiano. Una grammatica generativa in questo senso è un dispositivo formale che può enumerare ("generare") tutte e sole le frasi di un linguaggio. In un senso più stretto, una grammatica generativa è un dispositivo formale (o, in altri termini, un algoritmo) che può essere usato per decidere se una certa frase (o parola) è o meno grammaticale.

Nella maggior parte dei casi, una grammatica generativa è in grado di generare un numero infinito di "stringhe" (parole) a partire da un insieme finito di regole. Queste proprietà sono desiderabili per un modello del linguaggio naturale, poiché il cervello umano possiede una capacità finita, eppure gli esseri umani possono generare e comprendere una enorme quantità di frasi distinte

Formalmente una grammatica G è una quadrupla (N, Σ, S, P) composta da:

Σ**:** alfabeto dei simboli terminali;

N**:** alfabeto delle variabili o simboli non terminali;

$S \in N$**:** *simbolo iniziale* della grammatica;

P**:** insieme di coppie (v, w) di stringhe, dette *produzioni*, costruite sull'unione dei due alfabeti, denotate anche con $v \rightarrow w$.

La stringa v contiene almeno un simbolo non terminale.

Il *linguaggio* generato dalla grammatica è costituito da tutte le stringhe di terminali che possono essere ottenute partendo dal simbolo S ed applicando una produzione alla volta alle forme di frase via via prodotte.

Le grammatiche generative possono essere descritte e confrontate fra loro con l'aiuto della *Gerarchia di Chomsky*, proposta da lui negli anni 50. Questa definisce una serie di tipi di grammatiche formali aventi potere espressivo crescente.

Grammatiche regolari. Fra i tipi più semplici ci sono le grammatiche regolari (tipo 3). In una grammatica regolare:

- la parte sinistra delle produzioni è soltanto un simbolo non terminale
- la parte destra è vincolata nel seguente modo: può essere nulla (ε - stringa vuota), oppure nella forma a oppure nella forma aB, con a e B appartenenti rispettivamente ad N e a Σ (può cioè essere nulla, oppure un simbolo terminale seguito, eventualmente, da un solo simbolo non terminale).

Le grammatiche definite nel seguente esempio sono regolari destre (il non terminale nella parte destra delle produzioni, se c'è, segue il terminale).

Esempio.

1. La grammatica $\Sigma = \{a, b, c\}$, $N = \{S, A\}$ e $P = \{S \rightarrow aA, A \rightarrow bS|c\}$ genera stringhe del tipo *abc*, *ababc*, ... Il linguaggio da essa generato è $L = \{(ab)^n c,\ n > 0\}$ (una sequenza non vuota di "*ab*" seguita da una "*c*").
2. La grammatica $\Sigma = \{a, b\}$, $N = \{S, A, B\}$ e $P = \{S \rightarrow aA, A \rightarrow aA, A \rightarrow bB, B \rightarrow bB, B \rightarrow \varepsilon\}$ genera stringhe del tipo *abb*, *aaab*, ... Il linguaggio da essa generato è $L = \{a^n b^m,\ n, m > 0\}$ (almeno una "*a*" seguita da almeno una "*b*", con "*a*" e "*b*" in numero generalmente differente).

Automi a Stati Finiti non deterministici (ASFN). Nella teoria degli automi, un automa a stati finiti non deterministico è una macchina a stati finiti dove per ogni coppia stato-simbolo in input ci possono essere più stati di destinazione.

Formalmente, un automa a stati finiti non deterministico è una quintupla, (S, Σ, R, s_0, A) formata da

S**:** insieme finito di stati;

Σ**:** insieme finito dei possibili simboli in input, chiamato *alfabeto*;

$R \subseteq S \times (\Sigma \cup \{\varepsilon\}) \times S$**:** relazione di transizione;

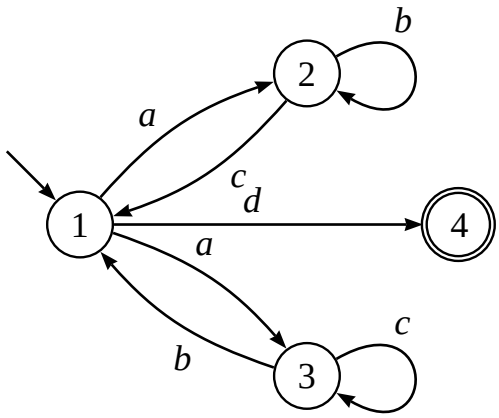
$s_0 \in S$**:** stato iniziale;

$A \subseteq S$**:** insieme di stati di accettazione.

Esempio. Il diagramma seguente rappresenta graficamente un ASFN con stati $S = \{1,2,3,4\}$, alfabeto $\Sigma = \{a,b,c,d\}$ e relazione di transizione

$$R = \{(1,a,2), (1,a,3), (2,c,1), (2,b,2), (3,b,1), (3,c,3), (1,d,4)\}$$

con stato iniziale 1 e finale 4.

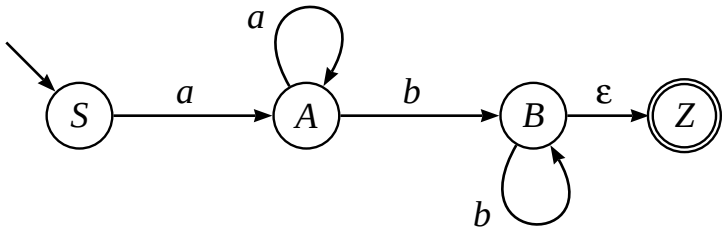


Da Grammatica Regolare ad ASF. Una grammatica regolare può essere trasformata in un automa a stati finiti mediante un semplice algoritmo. Le operazioni da effettuare sono:

1. si inserisce un nodo Z che corrisponde allo stato finale dell'automa;
2. si marca come nodo iniziale dell'automa quello corrispondente al simbolo iniziale della grammatica;
3. per ogni regola grammaticale, si introduce un arco nell'automa secondo le corrispondenze seguenti (U e A rappresentano simboli non terminali e t è un simbolo terminale):

$$\begin{aligned} U \to tA &\quad\Rightarrow\quad (U,t,A) \\ U \to t &\quad\Rightarrow\quad (U,t,Z) \\ U \to \varepsilon &\quad\Rightarrow\quad (U,\varepsilon,Z) \end{aligned}$$

Esempio. Data la grammatica $\Sigma = \{a,b\}$, $N = \{S,A,B\}$ e $P = \{S \to aA, A \to aA, A \to bB, B \to bB, B \to \varepsilon\}$ si ottiene l'automa seguente:



Riconoscimento di stringhe. In generale, per un ASFN M, un elemento di R viene detto transizione. Si dice che M riconosce una stringa di caratteri $s = b_1 \cdot \cdots \cdot b_k \in \Sigma^*$ se, a partire dallo stato iniziale s_0 e leggendo tutti i simboli di s uno alla volta, l'automa può transire in uno stato finale. Formalmente: $s = b_1 \cdot \cdots \cdot b_k$ è riconosciuto da M se esiste una sequenza di transizioni del tipo $(s_0, b_1, q_1), (q_1, b_2, q_2), \ldots, (q_{k-1}, b_k, q_k)$ con $q_k \in A$ e $q_1, \ldots, q_k \in S$. Per esempio, l'ASFN sopra rappresentato riconosce la stringa *abbb*, ma non *aa*.

Descrizione del progetto. Lo scopo di questo progetto è quello di realizzare un programma che:

1. accetti in ingresso una grammatica regolare (per semplicità si considerino grammatiche in cui produzioni del tipo $A \to \varepsilon$ non sono presenti);
2. costruisca l'ASFN corrispondente;
3. permetta di verificare se una o più stringhe, fornite in input dall'utente, sono riconosciute dall'automa o meno.

Le opzioni 1-3 devono essere fornite dall'utente tramite opportuno menù. Si tengano presenti le seguenti considerazioni.

- *Inserimento della grammatica:*
  - si richieda in modo esplicito l'inserimento del simbolo iniziale della grammatica
  - un possibile algoritmo per la gestione dell'inserimento delle produzioni è il seguente:
    1. inserire il non terminale per il quale si vogliono definire le produzioni, "0" per terminare l'inserimento della grammatica
    2. inserire una singola produzione (nella forma "*tV*" o "*t*") e premere invio, premere solamente invio per terminare l'inserimento di produzioni per il non terminale in questione
    3. se si è premuto invio senza inserire produzioni, ricominciare da 1, altrimenti ricominciare da 2.

    Al termine dell'inserimento della grammatica si dovrà verificare che non siano presenti non terminali che figurano a destra di produzioni ma non a sinistra (quindi verificare che per ogni non terminale utilizzato sia stata inserita almeno una produzione).

    Facoltativo: verranno apprezzati approcci diversi e più sofisticati di inserimento della grammatica, che possono prevedere anche controlli sintattici sulle produzioni inserite (ad esempio segnalare un errore nel caso in cui l'utente inserisca una produzione che non sia del tipo atteso - cioè qualcosa di diverso da t o tV, per t terminale e V non terminale).
- *Allocazione della memoria:* non essendo possibile limitare staticamente la dimensione della grammatica (insieme di non terminali, terminali e produzioni), per la costruzione dell'automa è necessario prevedere l'allocazione dinamica della memoria (v. `sbrk`).
- *Strutturazione:* il codice deve essere strutturato in procedure separate, ciascuna delle quali dovrà svolgere una funzione ben precisa.

- *Documentazione:* il codice deve essere commentato in modo significativo. Esempio di commento *non* significativo:

```
move $a0, $t1 # copia $t1 in $a0
```