Indice

La gerarchia di Chomksy vale per alfabeti di almeno due lettere.

1 Parola vuota ed ϵ-produzioni

Supponendo di avere una grammatica G di tipo 1, supponiamo di voler aggiungere la parola vuota al linguaggio generato. Se si aggiunge banalmente la regola per la parola vuota

$$S \to \epsilon$$

non ci sono garanzie che noi non abbiamo aggiunto anche altre stringhe al linguaggio, ad esempio nel caso

$$\alpha \Rightarrow \ldots S \ldots$$

inoltre perdiamo la proprietà della monotonicità delle forme sentenziali.

Quindi bisogna essere leggermente più delicati, dato $L = L(G)$, voglio costruire G' tale che $L' = L(G') = L(G) \cup \{\epsilon\}$. Per gare ciò definisco $G' = \langle V \cup \{S'\}, \Sigma, P \cup \{S' \to \epsilon \mid S\}, S' \rangle$.

Dal simbolo iniziale posso utilizzare $S \to \epsilon$ purché S non appaia su lato destro di nessuna produzione. Questo ovviamente vale anche per grammatiche di tipo 2 e 3.

Data una grammatica di tipo 2 che utilizza ϵ-produzioni

$$A \to \epsilon$$

è possibile trovare una grammatica di tipo 2 equivalente che genera lo stesso linguaggio, meno la parola vuota, che non utilizza ϵ-produzioni.

Anche nelle gramamtiche di tipo 3 possiamo ammettere ϵ-produzioni

$$A \to \epsilon$$

2 Automa a stati finiti

Un automa è un dispositivo che ha

- un nastro diviso in celle che contiene l'input, ogni cella contiene un simbolo dell'alfabeto Questo non può essere modificato.

- una testina che passa il nastro da sinistra a destra, per questo vengono chiamati anche automi *one* − *way*. Quando questa finisce il nastro deve poter rispondere se la parola appartiene o no al linguaggio. La testina contiene una macchina che ha un insieme finito di stati.

Un automa è quindi una quintupla $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, dove

- Q è l'insieme degli stati
- Σ è l'alfabeto finito di input
- δ è il programma dell'automa, o funzione di transizione
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è l'insieme degli stati finali

Ad ogni passo l'automa legge un simbolo, ed in base allo stato attuale e il simbolo in input sceglie il prossimo stato con δ

$$\delta : Q \times \Sigma \to Q$$

Definiamo

$$\delta^* : Q \times \Sigma^* \to Q$$

per induzione sulla lunghezza delle stringhe in input:

- $\forall q \in Q \mid \delta^*(q, \epsilon) = q$
- $\forall q \in Q, x \in \Sigma^*, a \in \Sigma \mid \delta^*(q, xa) = \delta(\delta^*(q, x), a)$

Visto che δ^* è effettivamente una estensione di δ, chiameremo δ^* δ.

Ora possiamo definire il linguaggio riconosciuto dall'automa $\mathscr{A}$ come

$$L(\mathscr{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$$

Ad esempio $\mathscr{A} = \langle Q = \{q_0, q_1\}, \Sigma = \{a, b\}, \delta, q_0, F = \{q_1\} \rangle$, con δ definita come

Per *aabb* abbiamo

$$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_0$$

quindi la stringa non è accettata perché $q_0 \notin F$. Questo è linguaggoi di tutte le stringhe con un numero dispari di b.

Invece che rappresentare δ come una tabella, di solito è comodo rappresentarla come un diagramma di transizione:

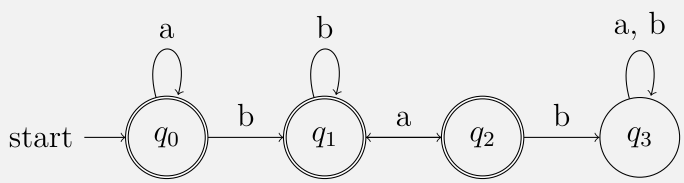
- Lo stato iniziale è rappresentato da una freccia entrante
- gli stati finali da un doppio cerchio.

Quindi una derivazione non è altro che un cammino in questo grafo, e la parola è accettata se si finisce in uno stato finale.

Dato $\Sigma = \{a, b\}$, e il linguaggio

$$\mathscr{L} = \{x \in \Sigma^* \mid P(x)\}$$

con P definita come la proprietà per cui tra ogni coppia di b successive in x c'è un numero di a pari.



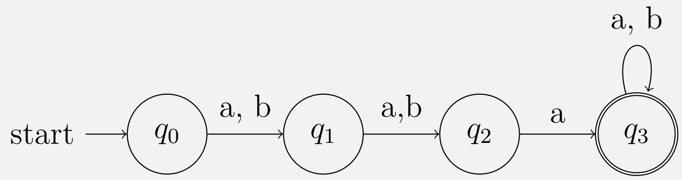
Visto che

$$\delta : Q \times \Sigma \to Q$$

nello stato q_2 bisogna aggiungere una freccia ad uno stato da cui non si può uscire, infatti se siamo in quello stato vuol dire che abbiamo trovato un numero dispari di a e una b, quindi la parola non è valida. Questo tipo di stati è detto stato trappola, e di solito vengono lasciati impliciti.

Dato $\Sigma = \{a, b\}$, e il linguaggio

$$\mathscr{L} = \{x \in \Sigma^* \mid \text{Il terzo simbolo di } x \text{ è una } a\}$$



Dato $\Sigma = \{a, b\}$, e il linguaggio

$$\mathscr{L} = \{x \in \Sigma^* \mid \text{ Il terzultimo simbolo di } x \text{ è una } a\}$$

Teniamo uno stato per ogni tripla di a, b, quindi $2^3 = 8$ stati

Una operazione che può interessare per le stringhe è costruire il linguaggio che riconosce le stringhe roversciate di uno. Dato un linguaggio $\mathscr{L}$, il linguaggio che riconosce le sue stringhe inverse è detto il reversal di $\mathscr{L}$. Nel caso dei due linguaggi precedenti se prendiamo l'automa del primo e lo invertiamo ottieniamo un automa valido (non deterministico) per il secondo.

2.1 Automi non deterministici

Definiamo un automa non deterministico come la quintupla $\mathscr{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, ma ora δ è definita come

$$\delta : Q \times \Sigma \to 2^Q$$

quindi δ non ritorna più un singolo insieme ma un insieme di possibili stati.

Una stringa viene accettata se esiste almeno un cammino che mi porta in uno stato finale. Estensiamo ancora δ alle stringhe

$$\delta^* : Q \times \Sigma \to 2^Q$$

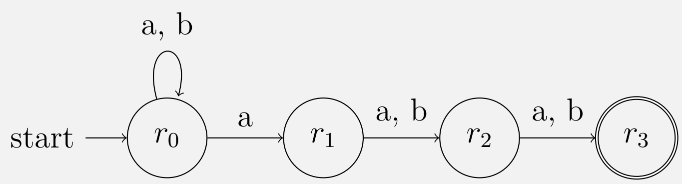
è definita come

- $\forall q \in Q \mid \delta^*(q, \epsilon) = \{q\}$
- $\forall q \in Q, x \in \Sigma^*, a \in \Sigma \mid \delta^*(q, xa) = \bigcup_{p \in \delta^*(q,x)} \delta(p, a)$

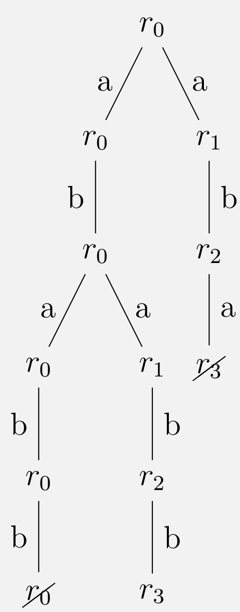
Come prima chiameremo δ^* semplicemente δ.

Ora il linguaggio accettato dall'automa $\mathscr{A}$ non deterministico è $L(\mathscr{A}) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \varnothing\}$.

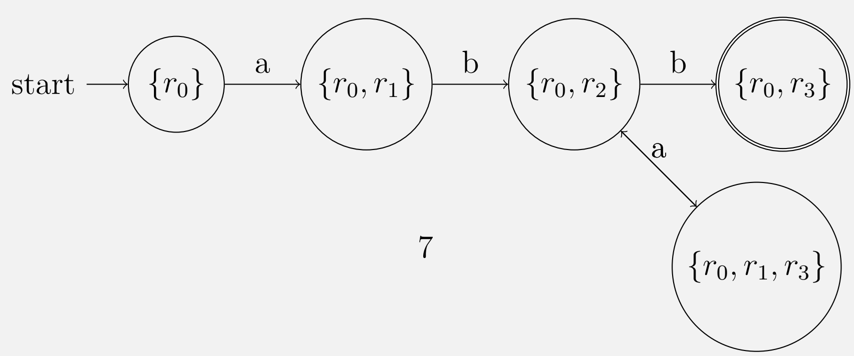
Dato l'automa non deterministico



con la stringha *ababb* mostriamo tutte le possibili strade che si possono prendere



è necessario che esista almeno una strada dell'albero che risponda sì per accettare. Questa struttura è chiamato albero di computazione, e un cammino descrive una computazione.
Bisogna supporre che l'automa riesca a sceglie la strada esatta. Alternativamente si può pensare che l'automa possa visitare tutte le strade in parallelo, quindi possiamo trasformare l'albero in



Questo

Ogni autma non deterministico (NFA) con n stati può sempre essere trasformato in un automa deterministico con al più 2^n stati. Nel caso degli automi a pila invece il modello non deterministico è più potente. Qui invece tra DFA e NFA non cambia la potenza computazionale, ma la semplicità descrittiva.