Automi e Linguaggi Formali

Sara Feltrin

26-02-18

Capitolo 1

Introduzione

Per iniziare, ci sono alcuni concetti di base da tenere a mente:

- **Alfabeto**: insieme finito e non vuoto di simboli, per esempio Σ = {0,1} oppure Σ = {a,b,c,d,e,..,z};
- **Stringa**: sequenza finita di simboli da un alfabeto Σ, per esempio: 011001 o abc;
- **Stringa vuota**: stringa con zero occorrenze di simboli dell'alfabeto Σ, denotata da ε;
- **Lunghezza di una stringa**: numero di simboli nella stringa, per esempio |w| denota la lunghezza della stringa w, quindi |01001| = 5;
- **Potenze di un alfabeto**: Σ^k insieme delle stringhe di lunghezza k con simboli da Σ, per esempio preso l'alfabeto Σ={0,1}: Σ^0={ε}, Σ^1={0,1}, Σ^2={00,01,10,11}. Viene chiamata potenza di un alfabeto poichè può essere vista come una potenza dove la base è il numero di simboli dell'alfabeto e l'esponente il numero della potenza dell'alfabeto (quindi, nell'alfabeto dei numeri binari con Σ^3, avremo $2^3 = 8$);
- **Insieme di tutte le stringhe**: per ottenere l'insieme di tutte le stringhe, usiamo il simbolo * e scriviamo $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup$ .. ;
- **Linguaggio**: dato un alfabeto Σ, chiamiamo linguaggio ogni sottoinsieme $L \subseteq \Sigma^*$ (compreso anche il linguaggio vuoto che non contiene nessuna parola).

Capitolo 2

Automi a stati finiti deterministici

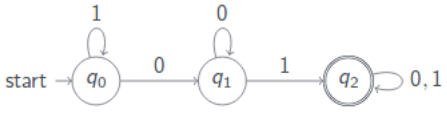
Un automa a stati finiti deterministico, chiamato anche DFA, è una quintupla $A = (Q, \Sigma, \delta, q_0, F)$ dove

- Q è un insieme finito di stati;
- Σ è un alfabeto finito, si intende quindi l'insieme di input che può leggere l'automa;
- δ è una funzione di transizione $(q, a) \mapsto q'$, ovvero dallo stato in cui sono, quando leggo il simbolo a, passo allo stato q';
- $q_0 \subseteq Q$ è lo stato iniziale dell'automa;
- $F \subseteq Q$ è un insieme di stati finiti;

L'automa può essere rappresentato sia come diagramma di transizioni sia come tabella di transizioni:

Esempio: costruiamo un automa A che accetta il linguaggio delle stringhe con 01 come sottostringa

- L'automa come diagramma di transizione:



- L'automa come tabella di transizione:

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

2.1 Linguaggio accettato da un DFA

La funzione di transizione prende in input uno stato e una parola dando in output una nuova parola. Definizione:

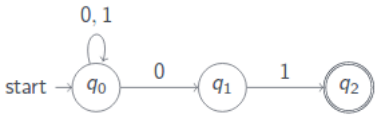
- **base**: $\delta(q, \varepsilon) = q \rightarrow$ ritorna lo stadio in cui è;
- **induzione**: $\widehat{\delta}(q, w) = \delta(\widehat{\delta}(q, x), a)$, dove $\widehat{\delta}$ rappresenta lo stato attuale e δ lo stato in cui mi troverò, a indica l'ultima lettera della parola che voglio leggere. NB: in $\widehat{\delta}$ faccio la ricorsione fino ad arrivare al caso base;

Detto ciò, possiamo definire il linguaggio accettato da A in questo modo: $L(A) = \{w : \widehat{\delta}(q_0, w) \in F\}$. Tutti i linguaggi accettati da DFA vengono chiamati **linguaggi regolari**.

Capitolo 3

Automi stati finiti non deterministici

È un automa che può trovarsi contemporaneamente in più stati diversi e le transizioni non devono per forze essere complete, per esempio:



Infatti questo non può essere un DFA perchè da q_0 se leggo 0 posso trovarmi contemporaneamente in q_0 e q_1, in più da q_1 posso muovermi solo in q_2 e da q_2 non posso proprio muovermi. Un automa a stati finiti non deterministici(NFA) è una quintupla $A = (Q, \Sigma, \delta, q_0, F)$ dove

- Q è un insieme finito di stati;
- Σ è un alfabeto finito;
- δ è una funzione di transizione che prende in input (q,a) e restituisce un sottoinsieme di Q;
- $q_0 \in Q$ è lo stato iniziale;
- $F \in Q$ è un insieme di stati finali;

Anche per i NFA abbiamo una definizione rigorosa:

- **base**: $\widehat{\delta}(q, \varepsilon) = q$;
- **induzione**: $\widehat{\delta}(q, w) = \bigcup_{p \in \delta(\widehat{q}, x)} \delta(p, a)$;

Data una parola, il nostro automa potrà trovarsi in uno dei tanti stati che siamo andati a calcolare.

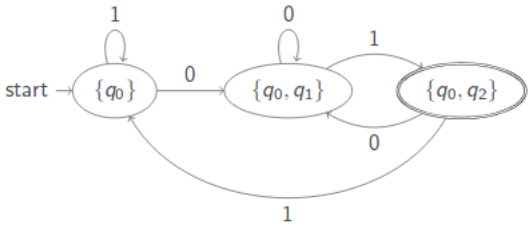
3.1 Equivalenza tra DFA e NFA

NFA e DFA sono in grado di riconoscere gli stessi linguaggi e l'equivalenza si dimostra mediante una **costruzione a sottoinsiemi**. Infatti, dato un NFA $N = (Q_N, \Sigma, q_0, \delta_N, F_N)$ costruiremo un DFA $D = (Q_D, \Sigma, q_0, \delta_D, F_D)$ tale che L(D)=L(N). Ogni stato del DFA, Q_D corrisponde ad un insieme di stati dell'NFA. *Uno stato del DFA F_D è finale se c'è almeno uno stato finale corrispondente all'NFA.* La funzione di transizione δ_D percorre tutte le possibili strade. Ad esempio:

Costruiamo δ_D per l'NFA qui sopra:

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

La tabella di transizione per D ci permette di ottenere il diagramma di transizione



Per semplificare il disegno, ho omesso gli stati non raggiungibili

Teorema 3.1.1. *Sia D il DFA ottenuto da un NFA N con la costruzione a sottoinsiemi. Allora L(D)=L(N).*

Teorema 3.1.2. *Un linguaggio L è accettato da un DFA se e solo se è accettato da un NFA.*

Capitolo 4

NFA con epsilon-transizioni

Le epsilon-transizioni vengono usate per muovere l'automa di stato anche se non viene dato nessun simbolo in input. Un automa a stati finiti non deterministico con ε-transizioni (ε-NFA) è una quintupla $A = (Q, \Sigma, \delta, q_0, F)$ dove cambia solo

- δ che è una funzione di transizione che prende in input uno stato in Q oppure un simbolo nell'alfabeto $\Sigma \cup \{\varepsilon\}$

L'eliminazione delle ε-transizioni procede per ε-chiusura degli stati, cioè prendendo tutti gli stati raggiungibili da q con una sequenza $\varepsilon\varepsilon...\varepsilon$. L'$\varepsilon$-chiusura viene indicata con **ECLOSE(q)**.

4.1 Equivalenza tra DFA e ε-NFA

Per ogni ε-NFA E c'è una DFA D tale che L(E)=L(D), e viceversa. Ogni stato del DFA corrisponde ad un insieme di stati chiuso per ε-chiusura. Uno stato del DFA è finale se c'è **almeno** uno stato finale corrispondente nell'ε-NFA.

Teorema 4.1.1. *Sia $D = (Q_D, \Sigma, S_0, F_D)$ il DFA ottenuto da un ε-transizioni E con la costruzione a sottoinsieme modificata. Allora L(D)=L(E).*

In altre parole, se riusciamo a dimostrare che hanno lo stesso risultato, allora i due automi sono equivalenti.

Teorema 4.1.2. *Un linguaggio L è accettato da un DFA se e solo se è accettato da un ε-NFA.*

Capitolo 5

Espressioni regolari

Diciamo che una parola è accettata oppure no in base allo stato in cui giungiamo alla fine di questa parola (se sono in uno stato finale, allora si tratta di una parola accettata dall'automa altrimenti no). Un'espressione regolare è un modo dichiarativo per descrivere un linguaggio regolare. Abbiamo diversi tipi di operazioni sui linguaggi:

- **Unione**: $L \cup M = \{w : w \in Loppurew \in M\}$
- **Concatenazione**: $L.M = \{uv : u \in Lev \in M\}$
- **Potenze**:
  - $L^0 = \{\varepsilon\}$
  - $L^1 = L$
  - $L^k = L.L...L(kvolte)$
- **Chiusura di Kleene**: $L^* = \bigcup_{i=0}^{\infty} L^i$

Inoltre le espressioni regolari sono costruite utilizzando

- un insieme di costanti di base:
  - ε per la stringa vuota
  - $\emptyset$ per il linguaggio vuoto
  - a,b,.. per i simboli $a, b, .. \in \Sigma$
- collegati da operatori:
  - $+$ per l'unione;
  - $\cdot$ per la concatenazione;
  - $*$ per la chiusura di Kleene;

Esistono anche delle **regole di precedenza** degli operatori:

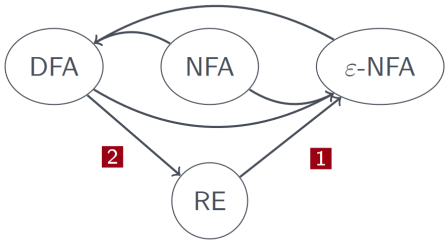
1 Chiusura di Kleene;

2 Concatenazione;

3 Unione.

5.1 Equivalenza tra FA e RE

DFA, NFA e ε-NFA sono tutti equivalenti.



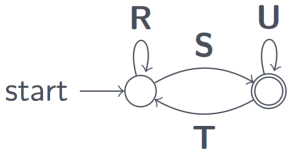
Infatti

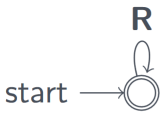
1 Per ogni espressione regolare **R** esiste un ε- NFAA, tale che L(A)=L(**R**);

2 Per ogni FA A possiamo costruire un'espressione regolare **R**, tale che L(**R**)=L(A).

5.2 Conversione per eliminazione di stati

Quando uno stato **q** viene eliminato, i cammini che passano per q scompaiono. Quindi si aggiungono nuove transizioni etichettate con espressioni regolari che rappresentano questi cammini eliminati. Alla fine otteniamo un'espressione regolare che rappresenta tutti i cammini dallo stato iniziale ad uno stato finale, ovvero **il linguaggio riconosciuto dall'automa**. Il procedimento è spiegato nel seguente algoritmo:

- Per ogni stato finale $q \in F$:
  1. elimina tutti gli stati tranne lo stato iniziale q_0 e q
  2. se $q \neq q_0$ l'automa finale è 

     che è equivalente a $(\mathbf{R} + \mathbf{SU}^*\mathbf{T})^*\mathbf{SU}^*$
  3. se $q = q_0$ l'automa finale è 

     che è equivalente a $\mathbf{R}^*$
- L'espressione regolare desiderata è l'unione $(+)$ di tutte le espressioni ottenute dalle regole 2 e 3

Capitolo 6

Grammatiche e linguaggi liberi dal contesto

Facciamo subito un esempio con un linguaggio non regolare, quindi libero dal contesto. Analizziamo il linguaggio dei palindromi con alfabeto {0,1} e per dimostrare che non è regolare usiamo il Pumping Lemma. Consideriamo la stringa palindroma $w = 0^n10^n$. Per il lemma possiamo scomporre w in w=xyz, in modo tale che y consista di uno o più 0 del primo gruppo. Di conseguenza, xz, che dovrebbero trovarsi in L_{pal} se questo fosse regolare, avrebbe meno 0 a sinistra dell'unico 1 rispetto a quello a destra. Dunque xz non può essere palindroma. Per definire il linguaggio L_{pal} possiamo usare una definizione ricorsiva:

- BASE: ε, 0 e 1 sono in L_{pal};
- INDUZIONE: se w è in L_{pal}, allora 0w0 e 1w1 sono in L_{pal}.

6.1 Grammatiche libere dal contesto

Il compito della grammatica con le sue regole è uno strumento per generare tutte le stringhe del linguaggio libero dal contesto. Definizione di grammatica libera da contesto:

1. Insieme finito T di simboli terminali che formano le stringhe del linguaggio (per L_{pal}, 0 e 1);
2. Insieme finito V di variabili (chiamate anche *non terminali*). Ogni variabile genera un linguaggio (per L_{pal} c'è solo la variabile P);
3. Un simbolo iniziale S che genera il linguaggio da definire (in L_{pal} coincide con P);
4. Un insieme finito (P o R) di produzioni o regole che hanno forma $testa \rightarrow corpo$;

La testa della produzione è una variabile che viene definita (parzialmente) dalla produzione mentre il corpo della produzione rappresenta un modo di formare stringhe nel linguaggio della variabile di testa ed è costituita da una stringa di zero o più terminali e variabili. Le stringhe si formano lasciando immutati i terminali e sostituendo ogni variabile del corpo con una stringa appartenente al linguaggio della variabile stessa.

Questi 4 componenti formano una grammatica libera dal contesto (CFG) che viene così rappresentata: $G = (V, T, P, S)$, dove V è l'insieme delle variabili, T i terminali, P l'insieme delle produzioni ed S il simbolo iniziale). Per quanto riguarda alla grammatica del linguaggio delle stringhe palindrome, avremo $G_{pal} = (\{P\}, \{0, 1\}, A, P)$ dove A rappresenta l'insieme delle produzioni delle palindrome:

1. $P \to \varepsilon$
2. $P \to 0$
3. $P \to 1$
4. $P \to 0P0$
5. $P \to 1P1$

Notazione compatta per descrivere le regole di produzione: $P \to \varepsilon|0|1|0P0|1P1$

6.2 Linguaggio di una grammatica

Se noi abbiamo una grammatica libera da contesto, questa definirà il linguaggio libero da contesto da cui deriva. Abbiamo due strade per passare dalla grammatica al linguaggio definita dalla grammatica stessa: dalla testa al corpo (**derivazione**) oppure dal corpo alla testa (**inferenza ricorsiva**). Sono due maniere opposte per vedere la grammatica definita, ma sono equivalenti tra loro.

Derivazione Viene indicata con $\Rightarrow$ ed indicata la relazione tra due forme sentenziali. Viene inoltre definita in questo modo:

Definizione 6.1 (Derivazione)**.** Sia G=(V, T, R, S) e sia αAβ dove α e β sono forme sentenziali ed A è in V, sia $A \to \gamma$ in R, allora αAβ

- $\Rightarrow \alpha\gamma\beta$
- $\Rightarrow \star$ è la chiusura riflessiva e transitiva della derivazione (significa zero o più passaggi)

Riprendendo l'esempio di G_{pal} dove $P \to \varepsilon|0|1|0P0|1P1$, abbiamo:

$P \Rightarrow 0P0 \Rightarrow 00P00 \Rightarrow 001P100 \Rightarrow 0010100$

$P \Rightarrow \star 0010100$