Appunti di Linguaggi e Computabilità

Domande e Risposte

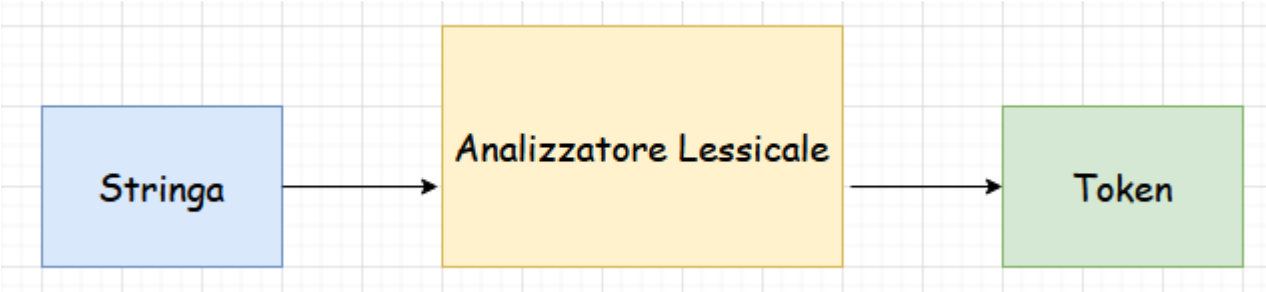
Capitolo 1 – Introduzione degli argomenti

1) Cosa si intende per analisi lessicale e analizzatore lessicale?

L'analisi lessicale è il processo che consiste nel ricevere in ingresso uno stream di caratteri (stringa) e produrre in output una sequenza di token, mentre l'analizzatore lessicale o lexer è un programma che si occupa di analizzare lessicalmente una stringa (esempio codice sorgente di un programma).

2) Quali sono gli input e gli output di un analizzatore lessicale?

L'analizzatore lessicale riceve in input una stringa (o stream di caratteri) e produce in output una sequenza di token.



3) Cosa si intende per token?

Il token è una coppia costituita da:

→ nome: simbolo astratto che rappresenta una categoria sintattica;

→ valore: sequenza di simboli del testo di ingresso.

Esempio:

```
<IDE,x1>
```

4) Cosa si intende per alfabeto?

Un alfabeto è un insieme finito e non vuoto di simboli che vengono identificati mediante la lettera maiuscola ∑ (sigma).

Esempio:

→ ∑ = {0,1}, l'alfabeto binario;

→ ∑ = {a, b, …, z}, l'alfabeto di tutte le lettere dell'alfabeto;

→ ∑ = {0, 1, …,9}, l'alfabeto delle cifre da 0 a 9.

5) Cosa si intende per stringa?

Una stringa è una sequenza finita di simboli scelti da un alfabeto.

Esempio: Sia l'alfabeto ∑ = {0,1} binario, le possibili stringhe sono:

→ w1 = 10110

→ w2 = 1101

→ w3 = 100101

6) Cosa si intende per stringa vuota?

La stringa vuota è una stringa particolare, estraibile sempre da ogni alfabeto che ha lunghezza 0. Generalmente è indicata dalla lettera greca ε (epsilon) o λ (lambda).

Osservazione

$|\varepsilon| = 0$

7) Cosa si intende per lunghezza di una stringa?

La lunghezza di una stringa è il numero di simboli che compongono la stringa che si indica con $|w|$.

Esempio: Sia l'alfabeto $\Sigma = \{0,1\}$ binario, le possibili stringhe sono:

→ $w1 = 10110$ e $|w1| = 5$;

→ $w2 = 1101$ e $|w2| = 4$;

→ $w3 = 100101$ e $|w3| = 6$;

Definizione formale (Induzione): Sia $|=| : \Sigma^+ \rightarrow N$, allora:

a) Base

$|\varepsilon| = 0$

$|a| = 1$, per ogni a appartenente a Σ

b) Ipotesi

se $w = a*x$ e "conosco" $|x|$ ($|x| = m$), allora $|a*x| = 1 + m$

8) Cosa si intende per potenza di un alfabeto?

Dati un alfabeto Σ e un numero n, si definisce potenza di un alfabeto Σ^n l'insieme delle stringhe estratte da Σ di lunghezza n.

Esempio:

→ $\Sigma^0 = \{\varepsilon\}$

→ $\Sigma = \{a, b\}$, $n = 2$ → $\Sigma^n = \{aa, ab, ba, bb\}$

Osservazione

Σ è l'insieme di tutte le stringhe ottenute da un alfabeto ed è infinito, ottenuto dall'unione di tutti gli Σ^n con n che varia da 0 a ∞.

9) Cosa si intende per concatenazione tra le stringhe?

Date due stringhe s_1 e s_2, si denota con s_1s_2 la nuova stringa risultante dalla concatenazione di s_1 con s_2, ovvero la stringa ottenuta appendendo in fondo a s_1 la stringa s_2.

Esempio: Data s_1 = ci e s_2 = ao, s_1s_2 = ciao

Osservazione

→ s_1s_2 /= s_2s_1
→ $|s_1s_2|$ = $|s_2s_1|$

10) Cosa è un linguaggio?

Il linguaggio è un insieme L di stringhe preso da un insieme ∑ dell'alfabeto.

Esempi di linguaggi

Dato ∑ = {0,1} costruisco il linguaggio tale per cui le stringhe sono composte da un numero uguale di 0 seguito da un numero uguale di 1.

Soluzione

Si scrive in maniera simbolica il linguaggio che si vuole costruire.

$$L = \{w \in \Sigma | w = 0^n 1^n, n \geq 0\}$$

Si ha che:
→ lunghezza minima è 0;
→ lunghezza massima non ha limiti.
Quindi

$$L = \{\epsilon, 01, 0011, 000111, \ldots\}$$

Problema sui linguaggi

Definizione: Dato un linguaggio L ed una stringa w definire se w appartiene o meno al linguaggio L.

Esempio di problema facile:

Dire se la stringa 'abba' appartiene al linguaggio delle stringhe palindrome ottenute da ∑ = {a, b}. La soluzione è un algoritmo che qualsiasi PC (anche il meno performante) riesce a risolvere.

Esempio di problema non troppo facile:

Dire se il numero 2034532126567 appartiene al linguaggio costruito da ∑ = {0, …, 9} tale per cui il numero rappresentato da w Є L è primo.

11) Definire la gerarchia di Chomsky

La gerarchia di Chomsky classifica i linguaggi in diverse categorie:
→ Ricorsivamente enumerabili: grammatiche di tipo 0, riconoscibili con le macchine di Turing;
→ Context – sensitive: grammatiche di tipo 1, riconoscibili con le macchine di Turing non deterministiche a nastro limitato.
→ Context – free: grammatiche di tipo 2, riconoscibili con gli automi a pila non deterministici.
→ Regolari: grammatiche di tipo 3, riconoscibili con gli automi a stati finiti.

12) Grammatiche Context – Free: definizione formale

Si definisce grammatica Context – free una n – upla G = (V, T, P, S) dove:
→ V: insieme delle variabili, esse verranno usate nelle regole di produzione, sono dette anche non – terminali;
→ T: insieme dei terminali, simboli che formano le stringhe del linguaggio;
→ P: insieme delle regole di produzione, la rappresentazione ricorsiva della definizione del linguaggio;
→ S: è il simbolo di partenza, una delle variabili di V.

Esempio: Prendiamo ancora il caso del linguaggio delle stringhe palindrome.

- $V_{ariabili} = \{V\}$
- $T_{erminali} = \{0, 1\}$
- $P_{roduzioni} = \{\dots\}$
- $S_{tart} = P$

$$G = (\{V\}, \{0, 1\}, P, V)$$
$$P = \{P \to \lambda, P \to 0, P \to 1, P \to 0P0, P \to 1P1\}$$

13) Definizione di Derivazione Left – Most e Right – Most

Una derivazione è detta sinistra (leftmost) se ad ogni passo di una derivazione la produzione è applicata al simbolo non terminale più a sinistra.
Una derivazione è detta destra (rightmost) se ad ogni passo di una derivazione la produzione è applicata al simbolo non terminale più a destra.

14) Definizione di Albero Sintattico

Sia G = (V, T, P, S) se w è la FRONTIERA o (PRODOTTO) di un ALBERO DI DERIVAZIONE. Un albero sintattico per G se e solo se:
→ per ogni nodo interno è etichettato con x Є V;
→ per ogni foglia è etichettato con a Є T;
→ ε è l'unico figlio del genitore;
→ se x Є V è un sottoalbero che produce $y_1, y_2, \dots, y_n$ Є V U T, allora x → $y_1, y_2, \dots, y_n$;
→ se x Є V è un sottoalbero che produce ε, allora x → ε Є P_G;
Esempio:
L = {w appartenente a {a, b, c, d}* | w = $a^m b^m c^n d^n$, m >= 0, n > 0}
S → XY
X → ε | aXb
Y → cYd | cd

aabbcccddd:
S → XY → aXbY → aaXbbY → aabbY → aabbcYd → aabbccYdd → aabbcccddd

15) Definizione di Frontiera
Sia L(G), un linguaggio definito dalla grammatica G, L(G) = {w Є T*| w è la frontiera di un albero di derivazione per G con radice S}.
Teorema: Sia A un albero sintattico di G con radice S e frontiera w, allora esiste una sola derivazione leftmost e una sola derivazione rightmost.

16) Grammatica di tipo 2: Context – Free
Tipologia di grammatica riconoscibile dagli automi a pila non deterministici le cui produzioni sono del tipo:
{x → B: x Є V & B Є (V U T) *}

17) Grammatica di tipo 0: Ricorsivamente enumerabili
Tipologia di grammatica che ingloba tutte le altre grammatiche e sono del tipo:
{a → B: a Є V & B Є (V U T) + & a contiene almeno una variabile & B Є (V U T) * }

18) Grammatica Ambigua
Sia G una grammatica, G è ambigua solo se esiste una stringa w generata da L(G) che ammette due alberi sintattici e due derivazioni left - most/right - most diverse. Per risolvere l'ambiguità si potrebbe:
→ cambiare grammatica mantenendo lo stesso linguaggio (ovvero le stesse stringhe);
→ un linguaggio è ambiguo se ogni grammatica che lo genera è ambigua.

La stringa viene interpretata in due modi diversi. Per risolvere tale ambiguità si potrebbe:
→ prima si moltiplica e poi si somma;
→ prima si somma e poi si moltiplica.
Esempio 1:
$L = \{w Є \{a, b, c\}^* \mid w = a^m b^n c^k, k > m + n \& m, n > 0\}$
Produzioni:
S → XY
X → aXc | aZc
Z → bZc | bc
Y → c | cY
Esempio 2:
$L = \{w Є \{a, b, c\}^* \mid w = a^{x+y} b^x c^y, x, y > 0\}$

Produzioni:
S → XY
X → aXc | aYc
Y → bYc | bc

19) Grammatica di Tipo 1: Context – sensitive

Si dice grammatica di tipo 1, la grammatica riconoscibile dalla macchina di Turing non deterministiche a nastro limitato. Le caratteristiche di tale grammatica sono:
→ sono dette context - dependent, cioè dipendenti dal contesto;
→ le produzioni sono tali per cui alla sinistra della produzione non hanno un'unica variabile, a1Aa2 → a1Ba2;
→ solo se la variabile A si trova "nel contesto" a1 ... a2, posso sostituirla con B, inoltre, se x1 → x2, allora |x1| ≤ |x2|;
→ sono due tipi:
1 – $a^n b^n c^n$
2 – $a^n b^m c^n d^m$

Esempio 1: Risoluzione tipo $a^m b^n c^m d^n$
→ costruisco $a^m C^m B^n d^n$, come tipo 2;
→ scambio le C con le B: $a^m B^n C^m d^n$
→ sostituisco B con b e C con c, in maniera controllata.
Le regole sono:
S → YZ
Y → aYC | aC
Z → BZd | Bd
Ora lavoro sul blocco interno, lasciando invariati gli estremi
CB → XB
XB → XC
XC → BC
Opero sulla sostituzione delle variabili
aB → ab
bB → bb
bC → bc
cC → cc

Esempio 2: Risoluzione tipo $a^n b^m c^n d^m$
→ raggruppare $a^n(BC)^n$ oppure $(AB)^n c^n$, in modo da ottenere una grammatica di tipo 2;

→ operare lo scambio tra B e C;
→ sostituire le variabili.

20) Grammatica di Tipo 3: Regolari

Le caratteristiche delle grammatiche regolari sono:
→ specificano il lessico di un linguaggio;
→ sono rappresentati da automi a stati finiti;
→ generano linguaggi regolari, rappresentati con una certa forma aritmetica;
→ è un modo alternativo per determinare come sono fatte le stringhe di un linguaggio;
→ le produzioni sono:
a) ε può comparire solo in $S \rightarrow \varepsilon$, altrimenti non compare;
b) tutte le produzioni sono solo lineari destre o solo lineari sinistre cioè
(sx) A → Ba | (sx) A → a oppure (dx) A → aB | (dx) A → a.
→ sia G una grammatica lineare sinistra , allora esiste G' lineare destra : L(G) = L(G') e viceversa.
→ se si impone una relazione, si esce dalle grammatiche di tipo 3 e si entra nelle grammatiche di tipo 2.

21) Operazioni tra i linguaggi

Le operazioni tra i linguaggi sono:
→ unione: $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L1 \cup w \in L2\}$
→ concatenazione: $L_1L_2 = \{w_1w_2 \in \Sigma^* \mid w_1 \in L_1 \& w_2 \in L_2\}$
→ potenza: $L^n = L * L * \dots * L$.
Esempi:

Dato il linguaggio
$L = \{w \in \{a,b,c,d\}^* \mid w = ab^m cd^n \text{ con } m, n \geq 0\}$
si costruisca la grammatica che lo genera.

Produzione Lineare Destra
S -> aB
B -> bB|cD
D -> dD|e

Produzione Lineare Sinistra
S -> Sd|Bc
B -> Bb|a

Dato il linguaggio
$L = \{w \in \{a,b,c,d\}^* \mid w = ab^m cd^n \text{ con } m \geq 0, n > 0\}$
si costruisca la grammatica che lo genera.

Produzione Lineare Destra
S -> aB
B -> bB|cD
D -> dD|d

Produzione Lineare Sinistra
S -> Sd|Dd
D -> Bc
B -> Bb|a

22) Espressione Regolare: definizione induttiva

Si definisce ricorsivamente RegEx su ∑:

Base:

0 Є ER

ε Є ER

Va Є ∑, a Є ER

Ipotesi Induttiva:

se E1, E2 Є ER, allora E1 + E2 Є ER, E1E2 Є ER, E1* Є ER, (E1) Є ER.

Per quanto riguarda i linguaggi si ha:

L (0) = 0

L (ε) = {ε}

L (a) = {a}

L (E1), L(E2)

L (E1 + E2) = L (E1) U L (E2)

L (E1E2) = L (E1) L (E2)

L (E1*) = (L (E1))*

L ((E1)) = L (E1)

Le proprietà principali sono:

→ E1 + E2 = E2 + E1;

→ E1E2 != E2E1

→ E + E = E;

→ EE != E;

→ E (F + H) = (EF) + (EH)

Esempio:
L = {1, 10, 110}
M = {ε, 01, 0111}
L ∪ M = {01, ε, 10}

23) Proprietà di chiusura dei LR

Dati determinati linguaggi regolari, se il linguaggio L viene formato per mezzo di determinate operazioni di tali linguaggi, allora anche L è regolare.
Le proprietà di chiusura sono:
1) unione: siano L e M linguaggi sull'alfabeto Σ, allora L ∪ M è il linguaggio che contiene tutte le stringhe che si trovano in L o in M oppure in entrambi.
2) complemento: sia L un linguaggio sull'alfabeto Σ, allora il complemento not(L), di L, è l'insieme delle stringhe in Σ* che non sono in L.
3) intersezione: siano L e M linguaggi sull'alfabeto Σ, allora L ∩ M è il linguaggio che contiene tutte le stringhe che si trovano sia in L sia in M.

24) Automi a Stati Finiti: definizione formale

Un automa a stati finiti è una quintupla $A = \langle Q, \Sigma, \Delta, I, F \rangle$, dove:
- Q è un insieme finito non vuoto di stati;
- Σ è un insieme finito non vuoto di simboli (alfabeto);
- $\Delta \subseteq Q \times \Sigma \times Q$ è la relazione di transizione;
- $I \subseteq Q$ è l'insieme di stati iniziali;
- $F \subseteq Q$ è l'insieme di stati finali.

Esempio: $\langle \{q1, q2\}, \{a\}, \{\langle q1, a, q2\rangle, \langle q2, a, q1\rangle\}, \{q1\}, \{q2\} \rangle$

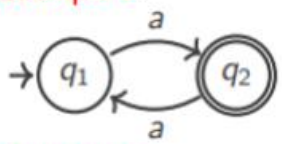
Gli elementi di un automa sono:
- un alfabeto (istruzioni);
- un insieme finito di stati (memoria);
- un insieme di regole di transizione (azioni);
- uno o più stati iniziali;
- stati designati come finali.

Le regole di transizione descrivono il nuovo stato della memoria in base all'istruzione. Dopo aver letto la sequenza, può finire in uno stato finale (accetta), o no (rifiuta).

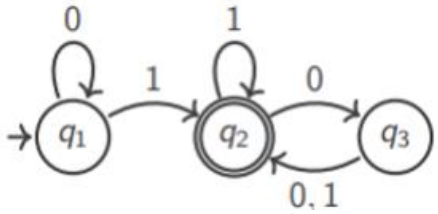
Un automa può essere rappresentato come un grafo etichettato $\langle Q, E, \ell \rangle$ con $\ell : E \to P(\Sigma)$. I nodi del grafo rappresentano gli stati, e gli archi le transizioni.
Gli stati iniziali si rappresentano con un semiarco e quelli finali con un doppio

bordo.

Esempio 1:



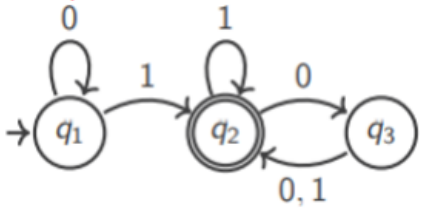
Esempio 2:



L'automa $\langle Q, \Sigma, \Delta, I, F\rangle$ dove

- $Q = \{q1, q2, q3\}$;
- $\Sigma = \{0,1\}$;
- $I = \{q1\}$;
- $F = \{q2\}$;
- $\Delta = \{\langle q1, 0, q1\rangle, \langle q1, 1, q2\rangle, \langle q2, 0, q3\rangle, \langle q2, 1, q2\rangle, \langle q3, 0, q2\rangle, \langle q3, 1, q2\rangle\}$;

Esempio 3:



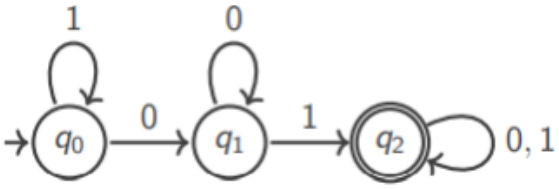
W = 1101

L'automa esegue la sequenza di stati $\langle q1, q2, q2, q3, q2\rangle$ e finisce sullo stato finale q2; quindi la accetta.

Rifiuta la sequenza W = 0010.

Esempio

$L = \{0,1\}^* \cdot \{01\} \cdot \{0,1\}^* = \{x01y \mid x, y \in \{0, 1\}^*\}$



$\langle\{q0, q1, q2\}, \{0, 1\}, \Delta, \{q0\}, \{q2\}\rangle$

25) Automi Non Deterministici: definizione formale

Si definisce NFA, è un automa che non prevede uno stato per ogni simbolo, ma prevede più stati per uno stesso simbolo:

→ è definito da $(Q, \Sigma, \delta n, q0, F)$

→ mentre la δn cambia: $\delta n: Q \times \Sigma \rightarrow P(Q)$.

26) ε – Automi Non Deterministici: definizione formale

Si dice ε – NFA, l'automa non deterministico che ammette l'utlizzo della stringa vuota ε. L'ε – mossa avviene senza leggere nulla ed è definito da $(Q, \Sigma, \delta_E, q0, F)$ e la funzione δ_E è definita δ_E: $Q \times \Sigma \cup \{\varepsilon\} \rightarrow P(Q)$.
La funzione δ_E' è definita δ_E: $P(Q) \times \Sigma \cup \{\varepsilon\} \rightarrow P(Q)$ come
→ $\delta'_E(0, a) = 0$;
→ $\delta'_E(S, a) =$ Uq appartiene ad S $\delta'_E(q, a)$;

27) ε – Chiusura: definizione formale

La funzione ε – Chiusura è una funzione definita come Eclose: $Q \rightarrow 2^Q$.
Definizione ricorsiva:
→ Base: Lo stato q appartiene a ECLOSE(q).
→ Induzione: Se lo stato p appartiene a ECLOSE(q) ed esiste una transizione, etichettata e, da p a r, allora r appartiene a ECLOSE(q). Più precisamente, sia δ la funzione di transizione di un e-NFA; se p appartiene a ECLOSE(q), allora ECLOSE(q) contiene anche tutti gli stati in δ(p, e).

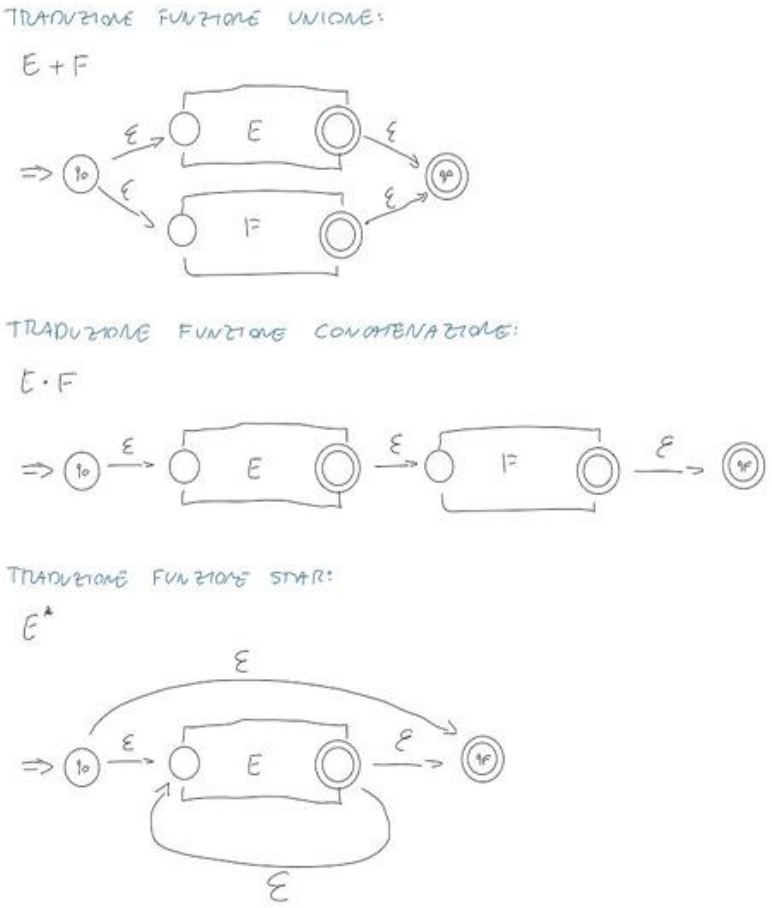
28) ε – Chiusura estesa: definizione formale (δ^)

Base: δ^(q, e) = ECLOSE(q). Se l'etichetta del cammino è e, si possono seguire solo archi etichettati e uscenti da q; questo è proprio il modo in cui ECLOSE viene definito.
Induzione: Suppongo ora che w = x * a, dove a è l'ultimo simbolo. Calcoliamo δ ^(q, w) come segue:
1) Pongo δ(q, x) = {p1, p2, … , pk}· In tal modo i sono tutti e soli gli stati raggiungibili da q lungo cammini etichettati x. Un tale cammino può terminare con una o più e-transizioni c può contenere altre e-transizioni.
2) Sia $U_{i=1}^{k}$ δ (P_i, a) l'insieme {r1, r2, …, rm}. In altre parole si seguono tutte le transizioni etichettate a dagli stati che si raggiungono da q con cammini etichettati X. Gli rj sono alcuni degli stati raggiungibili da q con cammini etichettati w. Gli altri stati raggiungibili si trovano seguendo f-archi uscenti dagli rj.
3) Infine δ(q, w) = $U_{j=1}^{m}$ ECLOSE (rj).

29) Trasformazione da NFA o da ε-NFA a DFA, e viceversa

a) Trasformazione DFA → ε-NFA
L'automa ε-NFA sarà costituito da:
→ q0 non deve avere archi entranti;
→ qF non deve avere archi uscenti e deve essere unico.

Per tradurre le funzioni (OR, STAR, AND) è necessario utilizzare l'algoritmo di Thomson.



b) Trasformazione ε-NFA → DFA

Si devono utilizzare due tabelle, una per la funzione delta ε-NFA e una per la funzione delta DFA. Queste tabelle avranno sulle righe gli stati e sulle colonne gli input. La seconda (quella del DFA) la riempio, mettendo come stati tutte le possibili combinazioni di stati dell'ε-NFA. Successivamente, per riempire questa tabella si considera lo stato (che può appunto essere un insieme di stati) e per ognuno dei suoi elementi si osserva in quale stato va e si effettua l'unione di tutti. Lo stato risultante è sempre un insieme di stati. Nella traduzione la tabella di output verrà impostata in modo tale che la prima riga rappresenti l'e-closure dello stato iniziale dell'ε-NFA. A questo nuovo stato iniziale si applica la funzione delta: per ogni input si ottiene (si noti che nella nuova tabella non abbiamo la ε mossa) un nuovo insieme di stati. Il processo di ottenimento di nuove righe della tabella si ripete finchè non ci saranno più nuovi insiemi di stati. L'insieme vuoto viene considerato come uno stato (stato pozzo).