Indice

op	DFA	NFA
unione	$n' \cdot n''$	$1 + n' + n''$
concatenazione	$n' \cdot 2^{n''}$	$n' + n''$
star	$1 + 2^{n'}$	$n' + 1$
intersezione	$n' \cdot n''$	$n' \cdot n''$
complemento	n'	$2^{n'}$

L'intersezione di automi funziona anche con NFA con la stessa costruzione dei DFA.

Date tre espressioni regolari

$$(a * b*)* \equiv (a + b)* \equiv b * (ab*)*$$

Sono tre linguaggi equivalenti.

Ci si può chiedere qual è il numero massimo di star innestate necessarie per generare un linguaggio. Introduciamo l'*altezza di start*, o star height h, come il numero massimo di star innestate. Definita nel seguente modo

$$h(E) = \begin{cases} 0 & \text{se } E = \varnothing \vee E = \epsilon \vee E \in \Sigma^* \\ \max(h(E'), h(E'')) & \text{se } E = E' \cdot E'' \vee E = E' + E'' \\ 1 + h(E') & \text{se } E = E'* \end{cases}$$

Definiamo poi la minima altezza per un linguaggio $L \subseteq \Sigma^*$ come

$$h(L) = \min\{h(E) \mid \ E \text{ denota } L \ \}$$

Teorema 1 (Dejan, Schutzenberger)**.**

$$\forall q > 0 \exists W_q \subseteq \{a, b\}^* \mid h(W_q) = q$$

Equesto è definito come

$$W_q = \{w \in \{a, b\}^* \mid \#_a(w) \equiv \#_b(w) \mod 2^q\}$$

La misura dei cicli è collegata al numero di cicli in un automa.

Per $q = 1, 2, \ldots,$ definiamo W_q ed abbiamo che

$$W_2 = ((ab + ba) + (aa + bb)(ab + ba)^*(bb + aa))^*$$

Nota 1. *Se* $|\Sigma| = 1$*, allora* $\forall L \subseteq \Sigma^* \mid h(L) \leq 1$.

1 Espressioni regolari estese

Supponiamo di avere il linguaggio

$$L = \{w \in \{a, b\}^* \mid \#_a(w) \text{ è pari } \wedge \#_b(w) \text{ è pari } \}$$

Costruiamo prima il linguaggio con solo le a pari

$$(b + ab * a)*$$

e con solo le b pari

$$(a + ba * b)*$$

intuitivamente il linguaggio L sarebbe, se avessimo l'intersezione,

$$(b + ab * a) * \cap (a + ba * b)*$$

Intoduciamo quindi le **espressioni regolari estese**.

Definiamo queste come le espressioni con, oltre alle operazioni di concatenazione, unione e star; l'intersezione ($\cap$) e il complemento (E^-). Introducendo il complemento possiamo ottenere l'intersezione attraverso l'unione.

Dato il linguaggio su $\Sigma = \{a, b\}$ delle stringhe con 3 a consecutive vogliamo riconoscere il suo complemento (senza 3 a consecutive). Questo diventa semplicemente

$$\overline{\overline{\emptyset} aaa \overline{\emptyset}}$$

Quindi ora senza usare la star possiamo esprimere anche linguaggi infiniti.

Si può definire il problema di trovare il l'altezza di star minima (eh) nelle espressioni regolari estese. Questo è un problema aperto. Si sa che valgono

$$\exists L \mid eh(L) = 0$$
$$\exists L \mid eh(L) = 1$$

Definiamo l'operazione di inversione o reversal. Data una stringa

$$w = a_1 a_2 \dots a_n$$

definiamo

$$w^R = a_n \dots a_2 \dots a_1$$

Per DFA e NFA basta invertire stati iniziali e finali e le transizioni. Però può accadere che un DFA invertito diventi non deterministico (e.g. stati finali multiplo diventano stati iniziali multipli, o più transizioni con lo stesso simbolo entranti in uno stato diventano più transizioni uscenti con lo stesso simbolo).

> Ad esempio il linguaggio il cui terzo simbolo è una a è un semplice DFA, mentre il linguaggio il cui terzultimo simbolo è una a ha 2^3 stati. In generale se chiediamo l'n-esimo simbolo da destra servono $n+1$ stati, mentre se chiediamo l'n-esimo simbolo da sinistra ne servono 2^n.

Mentre per le espressioni regolari, per la costruzione del reversal, basta solo invertire l'espressione. O più precisamente

- nei casi in cui $E = \varnothing \vee E = \epsilon \vee E \in \Sigma^*$, non cambia niente
- nei casi in cui $E = E' + E''$, abbiamo $E'^R + E''^R$
- nei casi in cui $E = E' \cdot E''$, abbiamo $E''^R \cdot E'^R$
- nei casi in cui $E = E'*$, abbiamo $(E'^R)*$

Definiamo l'operazione di *shuffle*

$$sh(x, y)$$

che produce ogni possibile combinazione dei suoi argomenti (più complicato di così, produce ogni possibile interpolazione, a non è mai dopo b e c non è mai dopo d), ad esempio

$$sh(ab, cd) = \{abcd, acdb, acbd, cabd, \dots\}$$

Dati due linguaggi, abbiamo che

$$sh(L', L'') = \bigcup_{x \in L', y \in L''} sh(x, y)$$

Supponiamo che $L' \subseteq \Sigma'^*$ e $L'' \subseteq \Sigma''^*$ e $\Sigma' \cap \Sigma'' = \varnothing$.

Definiamo l'operazione di quoziente tra due linguaggi

$$L_1 \setminus L_2 = \{x \in \Sigma^* \mid \exists y \in L_2 . xy \in L_1\}$$

Quindi le stringhe di L_1 in cui ho tolto un suffisso di L_2.

Ad esempio

$$L_1 = a + bc+$$
$$L_2 = bc + L_3 \qquad = c+$$

Ho che

$$L_1 \setminus L_2 = a+$$

e che

$$L_1 \setminus L_3 = a + bc*$$

Ad esempio da un linguaggoi su $\Sigma = \{0, 1\}$, voglio togliere tutti gli zeri finali

$$(L \setminus 0*) \cap ((0 + 1) * 1 + \epsilon)$$

Fatto 1. *Dato R linguaggio regolare ed L un linguaggio qualsiasi, allora*

$$R \setminus L$$

è ancora un linguaggio regolare.

Dimostrazione. Dato

$$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$$

l'automa per R.

Definisco

$$\mathcal{A}' = \langle Q, \Sigma, \delta, q_0, F' = \{q \mid \exists y \in L.\delta(q, y) \in F\}\rangle$$

come l'automa per $R \setminus L$.

Non è detto che sappiamo trovare una $y \in L$. Quindi questa dimostrazione non è costruttiva per tutti i linguaggi. ∎

1.1 Morfismo

Il linguaggio delle parentesi bilanciate (Dick) non è regolare. Dato un linguaggio di programmazione, lo possiamo ricondurre al linguaggio di Dick, ad esempio

$$\begin{aligned} \{ &\to (\\ \} &\to) \\ a &\to \epsilon \end{aligned}$$

Stiamo definendo un morfismo tra due alfabeti

$$h : \Sigma \to \Delta^*$$

In qenerale un morfismo su una strina è definito come

$$\begin{cases} h(\epsilon) = \epsilon \\ h(xa) = h(x)h(a) \end{cases}$$

e il morfismo su un linguaggio come

$$h(L) = \{h(w) \mid w \in L\}$$

Dati $\Sigma = \{a, b\}$, e $\Delta = \{0, 1\}$, definiamo il morfismo h come

$$\begin{aligned} h(a) &= 01 \\ h(b) &= 1 \end{aligned}$$

E vale che se abbiamo

$$L = ab*$$

questo corrisponde a

$$011*$$

E agli automi corrisponde

1.2 Sostituzione