Linguaggi Formali, Computabilità e Complessità

Agostino Dovier

Roberto Giacobazzi

Ugo Solitro

Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie 15
37134 Verona – Italy
E-mail address: solitro@sci.univr.it

Contenuti

CAPITOLO 9

Esercizi risolti

1. Linguaggi

ESERCIZIO 1.1. Definire matematicamente l'insieme di tutti i possibili codici fiscali.

SOLUZIONE. Siano $A = \{A, B, \ldots, Z\}$ l'insieme di tutte le lettere maiuscole e $C = \{0, 1, 2, \ldots, 9\}$ delle cifre decimali.

Il linguaggio dei codici fiscali CF è un sottoinsieme del seguente prodotto cartesiano.

$$A^6 \times C^2 \times A \times C^2 \times A \times C^3 \times A$$

Si noti che CF non coincide con l'insieme indicato: l'ultimo carattere è dipendente dai precedenti. □

ESERCIZIO 1.2. Descrivere usando il linguaggio matematico un sottoinsieme "sensato", ma molto semplificato del linguaggio naturale il cui "alfabeto" sia

$$\Sigma = \{\text{“Mario”, “Carla”, “vede”, “parla”, “gioca”, “con”, “insieme a”, “a palla”, }\}$$

SOLUZIONE. L'idea è quella di caratterizzare il ruolo possibile dei vari elementi.

Possiamo allora definire l'insieme delle persone, quello dei verbi transitivi, di quelli intransitivi eccetera:

$$P = \{\text{“Mario”, “Carla”, }\} \qquad T = \{\text{“vede”}\}$$
$$I = \{\text{“parla”, “gioca”}\} \qquad R = \{\text{“con”, “insieme a”}\}$$
$$M = \{\text{“a palla”}\}$$

Allora le frasi che si possono ottenere a partire dall'alfabeto Σ potrebbero essere quelle del seguente insieme:

$$(P \times T \times P) \cup (P \times I \times ((R \times P) \cup M) \cup (P \times I).$$

Sono sempre sensate le frasi che otteniamo? □

ESERCIZIO 1.3. Sia $\Sigma = \{a, b, c, \ldots, z\}$ l'insieme delle lettere minuscole dell'alfabeto latino.

Fra le parole riportate qui di seguito indicare quali appartengono a Σ^* e quali no motivando la risposta.

$$alfa, Gamma, zefiro, pratica, az3$$

SOLUZIONE. Le parole indicate appartengono Σ^* con l'esclusione della seconda (perché contiene una lettera maiuscola) e dell'ultima (perché contiene una cifra). □

ESERCIZIO 1.4. Se l'alfabeto $\Delta = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$; perché $0.1, -3$ non sono parole lecite?

SOLUZIONE. La prima contiene il punto, mentre la seconda contiene il segno. □

ESERCIZIO 1.5. Sia $\Sigma = \Delta \cup \{-\}$ (dove Δ è definito nell'esercizio precedente); caratterizzare l'insieme Z (linguaggio) delle rappresentazioni decimali dei numeri interi.

SOLUZIONE. Se definiamo $\Delta_+ = \Delta - \{0\}$ e $D = \Delta_+ \times \Delta^*$ allora l'insieme cercato è:

$$N = D \cup (\{-\} \times D).$$

□

ESERCIZIO 1.6. Sia data una stringa w di lunghezza $|w| = n$. Quanti sono i suoi prefissi, i suoi suffissi e le sue sottostringhe?

SOLUZIONE. Sia $w = a_1 a_2 \cdots a_n$ la stringa in questione; v è una **sottostringa** di w se esistono $i, j \in [0, \ldots, n]$ con $i \leq j + 1$ tali che $u = a_i a_{i+1} \cdots a_j$. Si noti che se $i = j + 1$ si conviene $u = \varepsilon$, la sottostringa impropria.

Allora i *prefissi* sono le sottostringhe del tipo $a_1 \cdots a_j$; dunque in tutto sono $n + 1$ sottostringhe al variare di j tra 0 e n.

Analogamente i *suffissi*, le sottostringhe con a_n come ultimo elemento, sono in numero di $n + 1$.

Per calcolare il numero totale delle sottostringhe contiamo quante sono le sottostringhe proprie (non nulle) con primo elemento a_k. Definiamo

$$W_k = \{\ w_{k,j} \mid w_{k,j} = a_k \cdots a_j,\ k \leq j\ \};$$

evidentemente la cardinalità di U_k dipende dal numero di valori che il parametro formale j può assumere, cioè $n - k + 1$.

Tenendo conto che l'insieme di tutte le sottostringhe è

$$W = \{\varepsilon\} \cup \bigcup_{k=1}^{n} W_k$$

segue che la cardinalità di W è così calcolata

$$|W| = 1 + \sum_{k=1}^{n} |W_k| = 1 + \sum_{k=1}^{n} (n - k + 1) = 1 + \sum_{h=1}^{n} h = 1 + \frac{n(n+1)}{2}$$

□

ESERCIZIO 1.7. Si provi che se $\Sigma \neq \emptyset$, allora Σ^* è numerabile.

Osservando che $\Sigma^* = \bigcup_{i \geq 0} \Sigma^i$ e che

$$(\Sigma^*) \qquad \begin{cases} \Sigma^0 &= \{\varepsilon\} \\ \Sigma^{n+1} &= \{ax : a \in \Sigma, x \in \Sigma^n\} \end{cases}$$

qual è la cardinalità di Σ^i ?

SOLUZIONE. Sia k la cardinalità dell'alfabeto Σ; dimostriamo per induzione prima cosa che la cardinalità di Σ^i è k^i.

Base: È immediato $|\Sigma^0| = |\{\varepsilon\}| = 1$.

Passo: È sufficiente osservare che esiste un isomorfismo di insiemi (biiezione) tra Σ^{n+1} e $\Sigma \times \Sigma^n$ da cui segue subito che

$$|\Sigma^{n+1}| = |\Sigma| \cdot |\Sigma^n| = k \cdot k^n$$

Essendo la famiglia $\Sigma_i (i \in \mathbb{N})$ costituita di insiemi disgiunti (perché?) ne segue che

$$|\Sigma^*| = \sum_{i \in \mathbb{N}} |\Sigma^i| = \sum_{i \in \mathbb{N}} k^i = \aleph_0$$

□

2. Automi a stati finiti

Esercizio 2.1. Rappresentare per mezzo di un automa a stati finiti il funzionamento di un ascensore in un edificio di tre piani.

Soluzione. Lo stato del dispositivo è rappresentato in modo naturale dal numero del piano in cui si trova l'ascensore; assumiamo quindi che l'insieme degli stati sia $Q = \{q_0, q_1, q_2\}$.

Il linguaggio dell'automa deve corrispondere alle azioni richieste da un utente: per semplicità assumiamo nell'ascensore ci siano solo tre pulsanti, uno per ogni piano, senza quello di *stop* e quello di *allarme*; il linguaggio sarà dunque $\Sigma = \{0, 1, 2\}$.

In questo caso particolare non ha senso parlare di stato iniziale e finale: il dispositivo funziona in continuazione senza arrestarsi.

La funzione di transizione può esser rappresentata dalla tabella seguente:

	0	1	2
q_0	q_0	q_1	q_2
q_1	q_0	q_1	q_2
q_2	q_0	q_1	q_2

oppure, sinteticamente, $\delta(q, n) = q_n$ per ogni n. □

Esercizio 2.2. Consideriamo gli automi descritti mediante la matrice di transizione (in tutti $F = \{q_1\}$).

($\mathcal{A}_1$)

	0	1
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_1	q_0

($\mathcal{A}_2$)

	0	1
q_0	q_2	q_0
q_1	q_2	q_0
q_2	q_1	q_2

Determinare in ambedue i casi il linguaggio accettato.

Soluzione.

L'alfabeto è $\Sigma = \{0, 1\}$, l'insieme degli stati $Q = \{q_0, q_1, q_2\}$.

Iniziamo con l'automa $\mathcal{A}_1$.

Osserviamo (se si disegna il grafo è più evidente) quanto segue:

- se si arriva nello stato q_1 (che è lo stato finale) si resta fermi in tale stato con qualunque *input*;
- a partire da qualunque stato il simbolo 0 porta nello stato finale q_1;
- se l'input inizia con una qualunque sequenza di 1 l'automa oscillerà tra gli stati non finali fino ad incontrare uno zero;
- se l'*input* non contiene 0, l'automa resta confinato in stati non finali.

Di conseguenza possiamo fare l'ipotesi che il linguaggio $L(\mathcal{A}_1)$ riconosciuto dall'automa sia l'insieme L_0 degli elementi di Σ^* contenenti almeno uno zero.

$L_0 \subseteq L(\mathcal{A}_1)$**:** Sia $w \in L_0$; allora esistono $w' \in \{1\}^*, w'' \in \Sigma^*$ tali che $w = w'0w''$. Allora $\delta(q_0, w') = q_0$ se $|w'|$ è pari, mentre $\delta(q_0, w') = q_2$ se $|w'|$ è dispari. La lettura di 0 porta l'automa nello stato q_1 e qualunque sia w'' l'automa resta nello stato finale.

$L(\mathcal{A}_1) \subseteq L_0$**:** Si dimostra la contronominale: *se w non contiene almeno uno* "0″ $w \notin L(\mathcal{A}_1$. Infatti, se w non contiene zeri a partire dallo stato iniziale q_0 non è possibile raggiungere lo stato finale q_1.

Per quel che riguarda l'automa $\mathcal{A}_2$...

□

ESERCIZIO 2.3. Si verifichi che i seguenti linguaggi, con $\Sigma = \{0, 1\}$, sono regolari:

(1) l'insieme di tutte le stringhe aventi tre 0 consecutivi;
(2) l'insieme di tutte le stringhe tali che il penultimo simbolo è 0.
(3) l'insieme di tutte le stringhe tali che il terzultimo simbolo è 0.
(4) l'insieme di tutte le stringhe tali che, se interpretate come numero intero (binario), sono divisibili per 2.
(5) l'insieme di tutte le stringhe tali che, se interpretate come numero intero (binario), sono divisibili per 4.
(6) l'insieme di tutte le stringhe tali che, se interpretate come numero intero (binario), sono divisibili per 5.

SOLUZIONE. Risolviamo solo alcuni dei casi richiesti.

(1) L'automa in questione deve "contare" tre zeri consecutivi; l'idea è che l'automa rimane nello stato iniziale finché non incontra uno zero, poi comincia a contare quanti sono e ritorna al punto di partenza se non sono in numero sufficiente. Sia allora $Q = \{\ q_0, q_1, q_2, q_3\ \}$ l'insieme degli stati, $F = \{\ q_3\ \}$ quello degli stati finali. La funzione di transizione è rappresentata dalla seguente tabella:

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q_3	q_3	q_3

(2) Intuitivamente in questo caso il simbolo zero deve portare l'automa in uno stato "prefinale", poi qualunque sia il simbolo successivo l'automa va nello stato finale; ma se l'input non è ancora consumato del tutto si ricomincia daccapo. Sia allora $Q = \{\ q_0, q_1, q_2\ \}$ l'insieme degli stati, $F = \{\ q_2\ \}$ quello degli stati finali. La funzione di transizione (l'automa non è deterministico!) è rappresentata dalla seguente tabella:

	0	1
q_0	q_0, q_1	q_0
q_1	q_2	q_2
q_2	$--$	$--$

(3) Il caso è analogo al precedente.
(4) Si tratta di determinare le stringhe binarie con ultima cifra zero. Allora, come nei casi precedenti, la tabella di transizione per un automa non deterministico è:

	0	1
q_0	q_0, q_1	q_0
q_1	$--$	$--$

(5) Come sopra: le due ultime cifre debbono essere 00.

(6) In questo caso la situazione è leggermente più complessa: la questione si sriduce ad individuare un criterio di divisibilità per 5 sulla base della rappresentazione binaria dei numeri e la sua traduzione in un automa.

□

ESERCIZIO 2.4. *(facoltativo)* Si determini il DFA equivalente al NFA:

	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_2\}$
q_2	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$

ove $F = \{q_2\}$. Qual è il linguaggio accettato?

SOLUZIONE. Una tecnica (quella illustrata a lezione) che consente di scrivere l'automa deterministico corrispondente è descritta dal procedimento che segue.

(1) Il nuovo insieme di stati è costituito dall'insieme delle parti dell'insieme Q originale degli stati con l'esclusione dell'insieme vuoto.
(2) La funzione di transizione è così definita: $\delta(Q', s) = \bigcup_{q \in Q'} \delta(q, s)$.
(3) Si può quindi procedere all'eliminazione di tutti quegli stati che risultano irragiungibili a partire dallo stato iniziale $Q_0 = \{q_0\}$

□

3. Linguaggi regolari

3.1. Espressioni regolari.

ESERCIZIO 3.1. Si considerino le seguenti identità tra espressioni regolari:

(1) $r + s = s + r$;
(2) $(r + s) + t = r + (s + t)$;
(3) $r(st) = r(st)$;
(4) $r(s + t) = rs + rt$, $(r + s)t = rt + st$;
(5) $\emptyset^* = \varepsilon$;
(6) $(r^*)^* = r^*$;
(7) $(\varepsilon + r)^* = r^*$;
(8) $(rs + r)^* r = r(sr + r)^*$;
(9) $(r + s)^* = r^* + s^*$;

Per ogni identità se ne verifichi la validità oppure si produca un contro-esempio.

SOLUZIONE. Con $[\![e]\!]$ indichiamo il linguaggio corrispondente all'espressione regolare e; allora $e = e'$ *se e solo se* $[\![e]\!] = [\![e']\!]$.

(1) $r + s = s + r$.
Se $[\![r]\!] = R$ e $[\![s]\!] = S$ allora, per la commutatività dell'operazione di unione fra insiemi,
$$[\![r + s]\!] = R \cup S = S \cup R = [\![s + r]\!]$$
(2) $(r + s) + t = r + (s + t)$
La dimostrazione è analoga alla precedente: questa volta si sfrutta l'associatività dell'unione.
(3) $r(st) = r(st)$.
Si tratta di dimostrare che l'operazione di concatenazione fra (i linguaggi rappresentati dalle) espressioni regolari è associativa; infatti:
$$[\![r(st)]\!] = [\![r]\!]([\![st]\!]) = [\![r]\!]([\![s]\!][\![t]\!]) = R(ST)$$
e inoltre
$$[\![(rs)t]\!] = ([\![rs]\!])[\![t]\!] = ([\![r]\!][\![s]\!])[\![t]\!] = R(ST)$$
Sinteticamente un elemento tipico di $R(ST)$ è della forma $a(bc)$ con $a \in R, b \in S, c \in T$; per l'associatività dell'operazione di concatenazione tra stringhe si ha che $a(bc) = (ab)c \in (RS)T$.
(4) $r(s + t) = rs + rt$, $(r + s)t = rt + st$.
Un elemento tipico in $[\![(]\!]r(s + t)$ è ab con $a \in [\![r]\!]$ e $b \in [\![s]\!] \cup [\![t]\!]$; ora se $b \in [\![s]\!]$ si ha che in particolare $ab \in [\![rs]\!] \subseteq [\![rs + rt]\!]$ e, analogamente, se $b \in [\![t]\!]$. Dunque
$$[\![(r(s + t)]\!] \subseteq [\![rs + rt]\!]$$
e in modo simile si verifica che
$$[\![rs + rt]\!] \subseteq [\![(r(s + t)]\!]$$
(5) $\emptyset^* = \varepsilon$.
Per definizione di L^* si ha che $\emptyset^0 = \{\varepsilon\}$; d'altra parte, per ogni $i > 0$ deve essere $\emptyset^i = \emptyset$.
(6) $(r^*)^* = r^*$.
Sia $a \in [\![(r^*)^*]\!]$; allora esiste $h \in \mathbb{N}$ tale che $a \in [\![(r^*)^h]\!]$ e perciò $a = a_1 a_2 \cdots a_h$ con $a_i \in [\![r^{k_i}]\!]$ per qualche $k_i \in \mathbb{N}$. Dunque $a \in [\![r^k]\!] \subseteq [\![r^*]\!]$ dove $k = \prod_{i=1}^{h} k_i$.

(7) $(\varepsilon + r)^* = r^*$.
Al solito si dimostra la doppia inclusione.
Se $a \in [\![(\varepsilon + r)^*]\!]$ allora esiste $h \in \mathbb{N}$ tale che $a \in [\![(\varepsilon + r)^h]\!]$, ovvero $a = a_1 a_2 \cdots a_h$ con $a_i \in [\![\varepsilon + r]\!]$, cioè $a_i = \varepsilon$ oppure $a_i \in [\![r]\!]$; da cui $a \in [\![r^k]\!]$ per $k \leq h$.
L'altro verso $[\![r^*]\!] \subseteq [\![(\varepsilon + r)^*]\!]$) è pressoché immediato.
(8) $(rs + r)^* r = r(sr + r)^*$. (tecnica analoga al precedente)
(9) $(r + s)^* = r^* + s^*$. (tecnica analoga al precedente)

□

3.2. Proprietà di chiusura e Pumping Lemma.

Esercizio 3.2. Si costruisca un ε-NFA che riconosce il linguaggio denotato dall'espressione regolare $((0 + 1)^*0011(0 + 1)^*)^*$.

Parte facoltativa

- Costruire il corrispondente DFA secondo le tecniche indicate nei relativi teoremi di equivalenza.
- Di quanti stati avete bisogno?
- Si cerchi di costruire direttamente un DFA che riconosce lo stesso linguaggio.

Soluzione. (traccia) La costruzione dell'automa può esser fatta meccanicanicmente seguendo le "istruzioni" del Teorema 2.1. La riduzione ad un DFA si può fare ricorrendo ai procedimenti descritti per il Teorema 6.1 e quindi per il Teorema 4.1. La difficoltà principale consiste nel tradurre le relative dimostrazioni in un procedimento schematico e chiaro.

Sorprendentemente (o quasi) la ricerca diretta di un DFA si rivela sostanzialmente più semplice. □

Osservazione 3.1. L'ultimo esercizio si presta ad una osservazione interessante.

A volte un procedimento molto generale, come quello descritto in una dimostrazione matematica, è molto semplice in astratto, ma porta delle notevoli complicazioni se applicato in pratica.

Potete trovare altri esempi di questa situazione?

Osservazione 3.2. **Rileggendo il Pumping Lemma**.

Il prossimo esercizio richiede l'applicazione del *Pumping Lemma* (Teorema 1.1).

Questo risultato, che risulta esser molto utile per lo studio dei linguaggi regolari, è spesso di difficile applicazione; la proprietà espressa dal *Lemma* è infatti relativamente complessa dal punti di vista logico-linguistico e una lettura affrettata porta rapidamente ad errori.

Indichiamo con $\mathcal{R}(L)$ la proprietà di un linguaggio di esser regolare e con $\mathcal{P}(L)$ la proprietà espressa dal *Lemma*, vale a dire:
Esiste una costante $n \in \mathbb{N}$ tale che per ogni $z \in L$, se $|z| \geq n$, allora per opportuni $u, v, w \in \Sigma^*$ valgono le seguenti proprietà:

(1) $z = uvw$,
(2) $|uv| \leq n$,
(3) $|v| > 0$,
(4) e per ogni $i \geq 0$, $uv^i w \in L$.

Iniziamo con alcune osservazioni preliminari.

- Il *Pumping Lemma* è schematizzato dall'implicazione seguente:

$$\mathcal{R}(L) \quad \Rightarrow \quad \mathcal{P}(L)$$

- La proprietà $\mathcal{P}$ è una condizione **necessaria**, ma **non sufficiente** perché un linguaggio sia regolare; dunque non basta verificare $\mathcal{P}$ per concludere che la regolarità del linguaggio!
- Esistono linguaggi che soddisfano la proprietà $\mathcal{P}$, ma che non sono regolari.
- Il *Pumping Lemma* invece è utile per dimostrare che un linguaggio non è regolare: infatti la contro-nominale[1] del *Lemma* è:

$$\neg\mathcal{P}(L) \quad \Rightarrow \quad \neg\mathcal{R}(L)$$

Osservazione 3.3. **Il gioco del Pumping Lemma**.

Un modo alternativo per "leggere" il *Lemma* è quello di vederlo come una contesa tra due giocatori: il primo *giocatore* propone un linguaggio per sostenere che non è regolare, mentre il secondo giocatore (detto *avversario*) cerca di dimostrare che il *Lemma* è soddisfatto.

Il gioco si svolge nel modo seguente:

(1) G: il (primo) giocatore propone un linguaggio L da esaminare;
(2) A: l'avversario propone un numero intero $n > 0$;
(3) G: preso atto che A ha fissato il numero n, il giocatore seleziona una stringa $z \in L$ (la scelta in generale dipende da n);
(4) A: l'avversario tenta di scomporre la stringa in tre parti $z = u \cdot v \cdot w$ tali che

$$|uv| \leq n, \quad \text{e} \quad |v| \geq 1$$

(5) G: il giocatore cerca un numero $i \geq 0$ che in generale dipende da n, u, v, w tale che $u \cdot v^i \cdot w$.

Se il giocatore riesce a trovare una strategia vincente (cioè un algoritmo che funzioni per tutte le possibili scelte dell'avversario) ha dimostrato che il linguaggio L non è regolare. In caso contrario vince l'avversario, il che non significa tuttavia che L sia regolare!

Per "rompere" un linguaggio (e dimostrare che non è regolare) bisogna mettersi nei panni del giocatore e trovare una strategia vincente. Si noti che non basta vincere con un esempio, ma bisogna esser in grado di vincere contro qualunque mossa dell'avversario.

Esercizio 3.3. Quali dei seguenti linguaggi sono regolari?

(1) $L_1 = \{0^{2n} : n \geq 1\}$;
(2) $L_2 = \{0^x 1^y 0^{x+y} : x \geq 1, y \geq 1\}$;
(3) $L_3 = \{0^p : p \text{ è primo}\}$.

Motivare la risposta.

Soluzione. Giochiamo?

(1) Non giochiamo perché il linguaggio è regolare e perderemmo!
Per convincersene basta verificare che l'automa in Figura 1
(2) Giochiamo!
E proponiamo di esaminare L_1.
A: l'avversario propone un certo $n > 1$;
non ci importa davvero quale numero sia: tanto noi dobbiamo esser in grado di fare la nostra contromossa qualunque sia il numero che ci viene proposto.
G: noi scegliamo la stringa $s = 0^n 1^1 0^{n+1}$:
lo scopo è di avere la sequenza iniziale di *zeri* lunga almeno n.

[1] Asserzione equivalente, ma in forma "negativa".

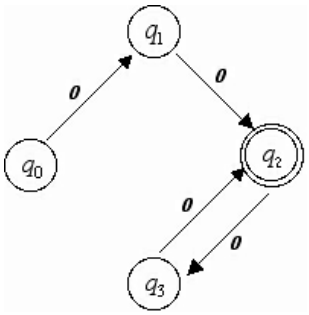

FIGURA 1. L'automa che accetta il linguaggio L_1.

A: l'avversario propone la decomposizione $s = uvw$;
siccome l'avversario deve rispettare le regole (le proprietà descritte per la decomposizione), noi sappiamo che

$$u = 0^i,\ v = 0^j,\ w = 0^h 1^1 0^{n+1}$$

con $i + j \leq n, j \geq 1$ e inoltre $0 \leq h \leq n-1$ in modo tale che $i + j + h = n$. Non c'è altro modo!

G: $s' = u \cdot v^2 \cdot w \notin L_1$!
Ormai abbiamo vinto! Infatti

$$s' = u \cdot v^2 \cdot w = 0^i 0^{2j} 0^h 1^1 0^{n+1}$$

ma $i + 2j + h + 1 = (i + j + h) + 1 + j = n + j \neq n$ perché $j \geq 1$.

(3) Qui il gioco si fà duro!
L'idea è abbastanza semplice: fissato il numero n, preso un numero primo $p > n$, se il linguaggio in questione fosse regolare (o quantomeno soddisfacesse la proprietà di *pumping*) si avrebbe che $p = a + b + c$ con $b > 0$ e per ogni $h \in \mathbb{N}$ $a + c + hb$ sarebbe un numero primo...
Ci possiamo credere? Allora si dimostri che non è possibile!

□

ESERCIZIO 3.4. Dati due DFA M_1 e M_2, si costruisca l'automa che riconosce $M_1 \cap M_2$ in maniera diretta.

Di quanti stati avete bisogno?

SOLUZIONE. (traccia) Un modo meccanico per trovare l'automa che serve è ricorrere alle proprietà di chiusura dei linguaggi regolari descritte nella Sezione 2.

Ma qui si chiede di costruire direttamente l'automa.

L'idea è quella di costruire un automa che esegua ambedue le verifiche "in parallelo". l'automa potrebbe avere come insieme di stati il prodotto cartesiano degli insiemi di stati dei singoli automi.

La funzione di transizione potrebbe allora essere "definita" così:

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)$$

Ovviamente abbiamo omesso un bel po' di dettagli... □

ESERCIZIO 3.5. Si dimostri che l'insieme delle stringhe di 0 e 1 tali che: non vi sono mai tre *zeri* consecutivi e non vi sono mai tre *uno* consecutivi, oppure vi sono almeno due *zeri* seguiti da due *uno* è un linguaggio regolare.

SOLUZIONE. (traccia della dimostrazione)
Si dimostra che i seguenti linguaggi su $\Sigma = \{0, 1\}$ sono regolari:

- L_1, insieme delle stringhe che contengono '000';
- L_2, insieme delle stringhe che contengono '111';
- L_3, insieme delle stringhe che contengono '0011'.

Per le proprietà di chiusura dei linguaggi regolari sono di conseguenza linguaggi regolari:

- L_1' linguaggio complementare di L_1;
- L_2' linguaggio complementare di L_2;

e di conseguenza

$$L = (L_1' \cap L_2') \cup L_3$$

che è il linguaggio richiesto. □

4. Linguaggi liberi dal contesto

Esercizio 4.1. Le espressioni nell'*aritmetica* a partire da un insieme di *costanti* $C = \{0\}$, quello delle funzioni $F = \{s, +, *\}$ e infine l'insieme delle variabili $V = \{x, x_0, x_1, \ldots\}$.

L'insieme E delle espressioni può essere definito secondo lo schema induttivo che segue:

- una costante è un'espressione lecita, ovvero $C \subseteq E$;
- una variabile è un'espressione lecita, ovvero $V \subseteq E$;
- se $e_1, e_2 \in E$ allora $s(e_1), e_1 + e_2, e_1 * e_2 \in E$.

Lo stesso insieme di espressioni può essere definito da una grammatica libera dal contesto.

(1) Definire l'alfabeto Σ necessario per la grammatica.
(2) Si tratta di un insieme finito?
(3) È possibile modificare la definizione induttiva delle espressioni in modo tale che l'alfabeto sia finito?
(4) Scrivere la grammatica.

Soluzione. Procediamo per punti:

- l'alfabeto può esser definito da $\Sigma = C \cup F \cup V \cup \{'(', ')'\}$;
- si tratta di un insieme infinito perché le variabili in generale costituiscono un insieme numerabile;
- è possibile definire Σ come linguaggio su di un alfabeto finito Γ; si consideri il linguaggio regolare $x1^*$: è sufficiente identificare le variabile x_i con l'espressione $x1^i \in [\![x1^*]\!]$; allora si può considerare come alfabeto
$$\Gamma = C \cup F \cup \{x, 1\} \cup \{'(', ')'\}$$
etc.
- Fissato Γ come alfabeto, il linguaggio libero dal contesto L_E delle espressioni aritmtiche è definito dalle seguenti produzioni:
$$\begin{aligned} E &\mapsto 0 \mid V \mid (E + E) \\ V &\mapsto x \mid xN \\ N &\mapsto 1 \mid 1N \end{aligned}$$
Si noti la somiglianza strutturale fra le produzioni e le regole induttive.

□

Esercizio 4.2. Un linguaggio di programmazione come il pascal non può essere descritto tramite un linguaggio regolare.

Ci sono "parti" del linguaggio pascal che costituiscono un linguaggio regolare?

Soluzione. La risposta è positiva.

Infatti gli *identificatori*, gli *interi*, le costanti sono definibili tramite espressioni regolari.

□

Esercizio 4.3. Rileggere la definizione della sintassi del linguaggio pascal cercando di scoprire se in tale contesto ci sono situazioni di ambiguità simili a quella descritta oppure come vengono risolte.

Soluzione. A rigore, ad esempio, un identificatore di costante è indistinguibile da un identificatore di variabile.

La differenza tra le due entità non è risolubile a livello sintattico. Tuttavia è vero che la definizione sintattica del linguaggio tende ad evidenziare l'aspetto semantico delle entità linguistiche. □

Bibliografia

[1] A. Dovier and R. Giacobazzi. Dispense per il corso di fondamenti dell'informatica, 2000.

[2] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computations*. Addison-Wesley, 1979.

[3] R. Péter. *Rekursive funktionen*. Akadémiai Kiadó, Budapest, 1953.

[4] Niklaus Wirth. *Algoritmi + Strutture Dati = Programmi*. Tecniche Nuove, 1987.