Approfondimento 3.6

Dimostrazione del Teorema 3.26.
Piuttosto che dare i dettagli della costruzione di s_G, faremo una serie di considerazioni che implicano l'esistenza di s_G, e che ci daranno nel contempo occasione di dare uno sguardo diverso alle grammatiche e ai linguaggi generati da esse.

Iniziamo con una digressione sulle definizioni ricorsive. Siamo forse abituati a vedere una definizione ricorsiva (cioè una definizione che usa nel corpo della definizione – nel *definiens* – il termine che si sta definendo – il *definiendum*) come un algoritmo. Ad esempio la definizione del fattoriale

$$\begin{array}{lcl} \mathit{fattoriale}(0) & = & 1 \\ \mathit{fattoriale}(n+1) & = & (n+1) \cdot \mathit{fattoriale}(n), \end{array}$$

può essere considerata come un algoritmo che, per ogni naturale n, permette il calcolo di $\mathit{fattoriale}(n)$, mediante lo "svolgimento" della ricorsione. Ma questo non è l'unico modo di intendere una definizione ricorsiva. Se non fossimo stati esposti al concetto di sottoprogramma ricorsivo, probabilmente vedremmo in una definizione ricorsiva una *equazione*, dove l'incognita è il *definiens*. La definizione del fattoriale è, in questo modo di vedere, una equazione tra funzioni, la cui soluzione, se esiste, è una funzione che soddisfa contemporaneamente le due clausole della definizione-equazione. Il nostro modo di vedere la definizione come un algoritmo è allora il procedimento risolutivo dell'equazione, che costruisce la funzione che soddisfa all'equazione.

Possiamo vedere una grammatica su un alfabeto di terminali Σ come una definizione ricorsiva. Anzi, come un sistema di equazioni ricorsive, nel quale le incognite sono i non terminali e le equazioni (le produzioni) sono costruite a partire da costanti (i terminali) e incognite mediante concatenazione e disgiunzione. Al sistema cerchiamo soluzione in Σ^*; in particolare, ci interessa la soluzione per una particolare incognita: il simbolo iniziale della grammatica. Questo modo di vedere le grammatiche è del tutto generale, e avremmo potuto condurre tutta la trattazione da questa prospettiva. Nel caso delle grammatiche regolari, però, la forma delle produzioni permette un percorso particolarmente semplice.

Iniziamo da un esempio semplicissimo, ma utile per il seguito. Consideriamo la grammatica regolare più semplice tra tutte quelle che hanno almeno una produzione "ricorsiva" (cioè nella quale compare il non terminale a destra):

$$A \to aA \mid b \mid c \mid \epsilon. \tag{16.1}$$

Coerentemente con quello che abbiamo detto, vogliamo considerare questa grammatica (cioè le sue produzioni) come un'equazione tra linguaggi. Quindi,

nel membro destro la concatenazione e la disgiunzione vanno ora viste come operazioni tra linguaggi, così come le abbiamo già considerate nel caso delle espressioni regolari. In altre parole, una soluzione a questa equazione è un insieme $L \subseteq \{a, b, c\}^*$ che, sostituito ad A, rende l'equazione un'identità, cioè per il quale

$$L = aL \cup \{b\} \cup \{c\} \cup \{\epsilon\}.$$

In generale vi saranno molte soluzioni; ad esempio $U = \{a, b, c\}^*$ è certo una soluzione, perché $aU = U$, e $U \cup \{b, c, \epsilon\} = U$. È possibile dimostrare che il linguaggio generato da una grammatica (secondo la definizione canonica che ne abbiamo dato nel Capitolo 2) è la soluzione *più piccola*, dove l'ordine inteso è quello di inclusione tra sottinsiemi.

Il linguaggio generato dalla grammatica (16.1) è

$$a^*(b|c|\epsilon),$$

come si vede sostituendo questa espressione al posto di A nella (16.1) e verificando che si ha davvero un'identità con semplici manipolazioni; partendo dal membro destro:

$$\begin{array}{rcl} aa^*(b|c|\epsilon) \mid b \mid c \mid \epsilon & = & (aa^*b \mid aa^*c \mid aa^*\epsilon) \mid b \mid c \mid \epsilon \\ & = & (a^+b \mid a^+c \mid a^+) \mid b \mid c \mid \epsilon \\ & = & a^*b \mid a^*c \mid a^* \\ & = & a^*(b \mid c \mid \epsilon) \end{array}$$

La cosa importante è che la soluzione $a^*(b|c|\epsilon)$ si ottiene in modo canonico dall'equazione (16.1): basta sostituire la ricorsione su A con la chiusura di Kleene e concatenarla con le alternative non ricorsive. Queste operazioni sono descritte da espressioni regolari a partire dalle costanti già presenti nella produzione.

Il caso immediatamente più complesso è quello di due non terminali, mutuamente ricorsivi. Con le solite convenzioni (maiuscole stanno per non terminali; minuscole per terminali), sia dunque G la grammatica con simbolo iniziale A e produzioni

$$\begin{array}{rcl} A & = & aA \mid bB \mid c \\ B & = & cA \mid aB \mid d. \end{array}$$

Se mettessimo $+$ al posto di $\mid$ sarebbe ancor più evidente che si tratta di un sistema lineare di due equazioni in due incognite. Sfruttando questa analogia (ma è ben più di una semplice analogia...), possiamo risolverlo "ricavando" B dalla seconda equazione, trattando per il momento A come una costante ed usando la tecnica già introdotta per la ricorsione semplice

$$B = a^*(cA \mid d).$$

Ora sostituiamo B nella prima equazione

$$A = aA \mid b(a^*(cA \mid d)) \mid c$$

e manipoliamola, usando le equazioni del riquadro di pagina 55, cercando di raccogliere A

$$A = aA \mid ba^*cA \mid ba^*d \mid c$$
$$A = (a \mid ba^*c)A \mid ba^*d \mid c$$

Siamo adesso nella forma semplice (una sola ricorsione) e dunque con la solita tecnica otteniamo la soluzione

$$A = (a \mid ba^*c)^*(ba^*d \mid c).$$

Osserviamo ancora una volta che la soluzione per A è data da un'espressione regolare, perché è ottenuta mediante manipolazione di espressioni regolari e sostituzione di definizioni regolari (l'espressione per B che abbiamo ricavato per prima) all'interno di altre espressioni regolari. In conclusione, la grammatica G definisce il linguaggio $\mathcal{L}[G] = \mathcal{L}[(a \mid ba^*c)^*(ba^*d \mid c)]$, che è regolare per costruzione.

Siamo ora in grado di trattare il caso generale. Una grammatica regolare con non terminali $\{A_1, \ldots, A_n\}$ e terminali in Σ può essere vista come un sistema lineare di n equazioni in n incognite

$$\begin{array}{rcl} A_1 & = & a_{11}A_1 \mid \cdots \mid a_{1n}A_n \mid b_{11} \mid \cdots \mid b_{1p} \\ A_2 & = & a_{21}A_1 \mid \cdots \mid a_{2n}A_n \mid b_{21} \mid \cdots \mid b_{2p} \\ & \vdots & \\ A_n & = & a_{n1}A_1 \mid \cdots \mid a_{nn}A_n \mid b_{n1} \mid \cdots \mid b_{np} \end{array}$$

dove tutti gli a_{ij} e b_{hk} sono elementi (non necessariamente distinti) di Σ. La soluzione a questo sistema può essere determinata attraverso la classica eliminazione gaussiana. Come nel caso del semplice esempio di due equazioni, si ricava un'incognita (per esempio A_n, supponendo che A_1 sia il simbolo iniziale) come espressione regolare costruita sulle costanti e sulle altre incognite (viste come costanti): $A_n = s_n[A_1, \ldots, A_{n-1}]$. Si sostituisce ora questa espressione al posto di A_n nella equazione per A_{n-1} e si ricava $A_{n-1} = s_{n-1}[A_1, \ldots, A_{n-2}]$ e così via, fino ad ottenere un'espressione per A_1 che non contiene più alcuna incognita: $A_1 = s_1$. Tutte le espressioni che si incontrano in questo procedimento sono espressioni regolari: la soluzione s_1 per A_1 sarà un'espressione regolare.

Abbiamo dunque giustificato il Teorema 3.26: data una qualsiasi grammatica regolare G, o accade che $\mathcal{L}[G] = \emptyset$ (e in questo caso non c'è nulla da dimostrare) oppure esiste un'espressione regolare s_G (la soluzione del sistema lineare per il simbolo iniziale della grammatica) che descrive il linguaggio generato dalla grammatica: $\mathcal{L}[G] = \mathcal{L}[s_G]$.

Come abbiamo già osservato, dobbiamo trattare a parte il caso del linguaggio vuoto, perché la nostra definizione di espressione regolare non contempla la possibilità di espressioni cui corrisponda il linguaggio vuoto. Grammatiche regolari e automi, invece, permettono naturalmente la definizione del linguaggio vuoto.

Esempio 16.6 La grammatica regolare con unica produzione $A \to aA$ definisce il linguaggio vuoto.

Esempio 16.7 Ogni automa (deterministico o non deterministico) con insieme di stati finali vuoto definisce il linguaggio vuoto.
Ogni automa nel quale gli stati finali non siano raggiungibili dallo stato iniziale definisce il linguaggio vuoto.