# Indice

# 1 Problema di Sakoda e Sipser – 1978

Questo è un problema irrisolto nella teoria degli automi two-way. La scorsa lezione abbiamo visto che

$$(a+b)^*a(a+b)^{n-1}$$

che per un 1DFA servono almeno $2^n$ stati, mentre per 1NFA servivano $n+1$ stati, infine abbiamo visto che per un 2DFA servivano $O(n)$ stati. Abbiamo visto anche risultati simili per il linguaggio

$$(a+b)^*a(a+b)^{n-1}a(a+b)^*$$

cioè per 1DFA almeno $2^n$ stati, per 1NFA $O(n)$ stati e per 2DFA $O(n)$.

Possiamo vedere che il 2DFA è deterministico, ma rimane lineare negli stati. Il problema di Sakoda – Sisper si interessa al costo delle simulazioni:

- da 1NFA a 2DFA
- da 2NFA a 2DFA

La migliore trasformazione da 1NFA a 2DFA è la costruzione ai sottoinsiemi, che però genera un 1DFA, infatti non si conosce una trasformazione che generi 2DFA. La trasformazione da 2NFA a 2DFA è la trasformazione a DFA, che però ha complessità $2^{O(n^2)}$. Quindi entrambe le trasformazioni hanno complessità esponenziale.

Il lower bound da 1NFA a 2DFA è $n^2$, quindi il gap tra upper bound e lower bound è grande.

Sakoda e Sipser hanno congetturato che la trasformazione non sia polinomiale. Sakoda e Sisper hanno mostrato che ci sono delle famiglie di linguaggi per cui se risolvo il problema per quella famiglia lo risolvo per tutti (???). Similmente al problema $P \stackrel{?}{=} NP$.

Invece si sa che se si vuole passare da 1NFA a 2DFA sweeping, il lower bound è esponenziale. Mentre si sa anche che da 2DFA sweeping a 2DFA c'è sempre un lower bound esponenziale.

È stato studiato anche il caso di linguaggi unari, cioè linguaggi il cui alfabeto di input ha una sola lettera. Si è mostrato che passare da 1NFA a 2DFA ha lower bound polinomiale, mentre nel secondo caso il problema è difficile.

# 2 Tecniche di dimostrazione che un linguaggio non sia regolare

Abbiamo già visto come strumento la distinguibilità, per cui se per un linguaggio si riesce a costruire un insieme di stringhe infinito tra loro distinguibili, allora il linguaggio non può essere regolare.

Alternativamente possiamo utilizzare le operazioni di chiusura, se dato un linguaggio la sua chiusura non è regolare, allora anche l'originale non lo è. Ad esempio dato

$$L = \{a^n b^n \mid n \geq 0\}$$

questo è ottenibile da

$$L' = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$$

facendo

$$L = L' \cap a^* b^*$$

Visto che $a^* b^*$ è regolare, ma $L$ non è regolare, allora $L'$ deve non essere regolare. Ora che sappiamo che $L'$ non è regolare, il linguaggio

$$L'' = \{w \in \{a, b\}^* \mid \#_a(w) \neq \#_b(w)\}$$

non è regolare in quanto complemento di $L'$.

**Lemma 1** (Pumping lemma)**.** *Sia $L$ un linguaggio regalare, allora esiste una costante $n \geq 0$, tale che per ogni $z \in L$ con $|z| \geq n$, che può essere scomposta in tre parti*

$$z = uvw$$

*tali che $|uw| \leq n$, $v \neq \epsilon$ e*

$$\forall k \geq 0 \mid uv^k w \in L$$

*Dimostrazione.* Dato $\mathcal{A}$ un automa per $L$, questo ha $n$ stati. Se la stringa riconosciuta ha più di $n$ simboli, allora devono esistere $i, j$ con $i < j$, per cui $q_i = q_j$. ■

Si chiama pumping lemma perché permette di gonfiare una stringa. Il teorema è interessante per sé, il lemma è utile per dimostrare altro, quindi i lemmi di solito sono finalizzati alla dimostrazione di un teorema.

Sia

$$L = \{a^n b^n \mid n \geq 0\}$$

dimostriamo che per qualsiasi costante $N$ data, ci sarà almeno una stringa del linguaggio di lunghezza $\geq N$, per cui qualunque decomposizione non fa valere una delle condizioni del pumping lemma. Priandiamo ad esempio la stringa

$$z = a^N b^N = uvw$$

visto che $|uv| \leq N$, sappiamo che $u \in a^*$ e $v \in a^+$, e $w \in a^* b^N$, cioè esistono $t, s$ con $t \geq 0, s > 0$, per cui

$$u = a^t$$
$$v = a^s$$
$$w = a^{N-(t+s)} b^N$$

Ora prendiamo

$$uv^o w = a^{N-s} b^N$$

ma questo ha meno $a$ che $b$, quindi non è nel linguaggio.

Dato

$$L = \{a^{2^n} \mid n \geq 0\}$$

supponiamo che $L$ sia regolare, e sia $N$ la costante del pumping lemma. Per comodità scegliamo

$$z = a^{2^n}$$

con $2^n > N$. E scomponiamola nelle tre parti $z = uvw$. Ora $v = a^i$, con $1 \leq i \leq N$, per la seconda condizione e la prima condizione del pumping lemma rispettivamente. Data una stringa generica

$$|uv^k w| = 2^n + (k-1) \cdot i$$

per $k = 2$ abbiamo $2^n + i$, che utilizzando le limitazioni su $i$ sappiamo che

$$2^n + 1 \leq 2^n + i \leq 2^n + N < 2^n + 2^n = 2^{n+1}$$

quindi $uv^2k \notin L$.

Il pumping lemma è una condizione necessaria, vediamo ora un esempio di un linguaggio che soddisfa la condizione, ma che mostremo (esercizio) che non è regolare.

$$L = \{a^m b^n c^n \mid m \geq 1, n \geq 0\} \cup \{b^m c^n \mid m \geq 0, n \geq 0\} \subseteq a^* b^* c^*$$

quindi se la parola inizia per $a$ chiediamo che le $b$ e le $c$ siano in egual numero. Data una stringa

$$z = a^m b^* c^*$$

con $m > 0$, possiamo sempre scrivere $z = uvw$ con $uv \in a^+$, e in questo modo riusciamo sempre a stare nel linguaggio.

Dato un linguaggio regolare vogliamo sapere se questo sia vuoto, finito, infinito.

**Lemma 2.** *Sia $L$ un linguaggio regolare, ed $N$ la costante del pumping lemma per $L$.*

1. *$L \neq \varnothing$ sse $L$ contiene almeno una stringa $z$ con $|z| < N$*
2. *$|L| = \infty$ sse $L$ contiene almeno una stringa $z$ con $N \leq |z| \leq 2N$*

*Dimostrazione.*

- ($\Leftarrow$, 1) $L$ contiene almeno una stringa $z$ con $|z| < N$ allora è ovvio che $L$ non è vuoto

- ($\Rightarrow$, 1) sia $L \neq \varnothing$, e sia $z$ la stringa più corta in $L$, allora
  - se $|z| > N$ allora ho finito
  - se $|z| \geq N$, allora per il pumping lemma la posso scomporre in $z = uvw$ con $v \neq \epsilon$ e $z' = uw \in L$, quindi $|z'| < |z|$, quindi $z$ non era la più corta
- ($\Leftarrow$, 2) sia $z \in L$ con $|z| \geq N$, allora $z = uvw$ con $v \neq \epsilon$, allora posso usare il pumping lemma per costruire infinite stringhe nel linguaggio $uv^k w$
- ($\Rightarrow$, 2) sia $z$ la più corta stringa di lunghezza almeno $N$ in $L$, allora
  - se $|z| < 2N$, ho finito
  - se $|z| \geq 2N$, allora scompongo $z = uvw$ con $v \neq \epsilon$ e $z' = uw \in L$. Ora la lunghezza di $z'$ è

    $$|z'| = |z| - |v|$$

    con $1 \leq |v| < N$. Visto che al massimo cancello $N$ simboli e $z$ era di lunghezza almeno $2N$, quindi $z'$ ha lunghezza almeno $N$, e allora siamo arrivati ad un assurdo visto che $z$ non è la più corta stringa di dimensione almeno $N$

■

Dato un linguaggio $L$ regolare voglio sapere

- se $L$ è vuoto, posso usare la prima condizione del lemma precedente, provando tutte le stringhe da 0 a $N$
- se $L$ è finito, posso usare la seconda condizione del lemma precedente, provando tutte le stringhe di dimensione da $N$ a $2N$
- se $L$ è infinito, come sopra

Questo è poco efficiente.

Un modo più efficiente per vedere se un linguaggio non è vuoto è vedere se esiste un cammino da uno stato iniziale ad uno finale. Per vedere se un automa è finito o infinito basta vedere se ci sono cicli in un cammino da uno stato inizale ad uno finale.

Dato $L$ per capire se $L \stackrel{?}{=} \Sigma^*$, basta vedere se il suo complemento è vuoto.

$$L = \Sigma^* \Leftrightarrow L^c = \varnothing$$

equivalentemente l'automa minimo deve ridursi ad un solo stato finale.

Dati due linguaggi regolari $L_1$ e $L_2$ per capire se $L_1 \stackrel{?}{\subseteq} L_2$.

$$L_1 \subseteq L_2 \Leftrightarrow L_2 \setminus L_1 = \varnothing$$

dove la differenza è definita come $L_1 \cap L_2^c$. Possiamo definire l'automa per la differenza similmente all'automa prodotto, ma accettare solo se la prima componente è in uno stato finale e la seconda no.

Dati due linguaggi regolari $L_1$ e $L_2$, per capire se $L_1 \stackrel{?}{=} L_2$, basta usare il risultato di sopra per l'inclusione

$$L_1 = L_2 \Leftrightarrow L_1 \subseteq L_2 \wedge L_2 \subseteq L_1$$

Alterantivamente si può definire ancora un automa simile al prodotto, in cui gli stati raggiungibili sono solo quelli in cui entrambi gli stati sono finali o no. Ancora visto che gli automi minimi sono unici, si può vedere se un automa minimo è isomorfo all'altro.