Esercizi risolti su linguaggi

Nicola Gulmini

- $L_1 \notin Reg$, $L_2 \in Reg$ allora $L_1 \cap L_2$ può essere non regolare.

  Vero, è sufficiente fornire un esempio: $L_2 = \{a^*b^*\}$ e $L_1 = L_{\neq}$ allora $L_{\neq} \cap L_2 = L_{=}$.

  Consiglio: nei casi di intersezione e unione l'uso di Σ^* e $\varnothing$ si rivela spesso molto utile. Nei casi di concatenazione si ricordi che $\varnothing$ è un annichilitore: $L \cdot \varnothing = \varnothing \ \forall L$.

  In questo caso, infatti, per fare prima è sufficiente prendere un qualsiasi linguaggio non regolare $L_1 \subseteq \Sigma^*$ e intersecarlo con Σ^*, che è regolare, per ottenere sempre L_1.

- L'intersezione tra un linguaggio finito e un linguaggio regolare è sempre regolare.

  Vero: l'intersezione tra un linguaggio finito e un qualsiasi altro linguaggio dà sempre un linguaggio finito, che è regolare.

- Ogni sottoinsieme di un linguaggio regolare è regolare.

  Falso, basta pensare che tutti i linguaggi, anche i non regolari, sono sottoinsiemi di Σ^* che è regolare.

- Ogni sottoinsieme di un linguaggio non regolare è non regolare.

  Falso. Una stringa di un linguaggio non regolare è da sola un insieme di cardinalità 1, finito e quindi regolare, per il quale si può costruire un automa ad hoc.

  Si può pensare anche all'insieme vuoto, sottoinsieme di tutti i linguaggi e finito perché contiene zero stringhe.

- Se togliamo un numero finito di stringhe a un linguaggio regolare, il linguaggio ottenuto è ancora regolare.

  Vero. L'enunciato dice che facendo la differenza insiemistica tra un regolare e un linguaggio finito il linguaggio ottenuto è ancora regolare,

e ciò è garantito dalla chiusura della classe Reg rispetto alla differenza insiemistica.

- La concatenazione di un linguaggio regolare e uno non regolare può fornire un linguaggi regolare.

  Vero. Concatenando un qualsiasi $L \notin Reg$ e $\varnothing \in Reg$ si ottiene $\varnothing$ per annichilazione.

- L finito, allora $L' = \{w \cdot w^R |\ w \in L\}$ è regolare.

  Vero, perché anche L' è finito. In particolare $|L'| = |L|$.

- $L \in Reg$, allora $L' = \{w \cdot w^R |\ w \in L\}$ è regolare.

  Qui bisogna prestare attenzione alla definizione del linguaggio in forma intensiva: il descrittore dell'insieme ha effetto solo *localmente*, perciò non è vero che $L' = L \cdot L^R$, come si sarebbe portati a credere. La prova è immediata: supponiamo che le stringhe 010 e 110 $\in L$, allora $010 \cdot 011 \in L \cdot L^R$ ma $\notin L'$.

  L'enunciato è falso, usando il pumping lemma lo si può facilmente dimostrare.

- $\bar{L}$ finito, allora $L' = LL^R$ è regolare.

  Vero. $\bar{L}$ finito $\Rightarrow$ regolare. Sia $L^C = \bar{L}$. Data la chiusura rispetto alla complementazione si ha $\bar{L}^C = L \in Reg$. Siccome la classe dei linguaggi regolari è chiusa anche rispetto al reverse e alla concatenazione, $L' \in Reg$.

- $L_1 = \{a^n b^n |\ n \geq 1\}$ (che sappiamo essere $\notin Reg$), $L_2 = \{a, b\}^*$ allora $L_3 = L_2 \cdot L_1 \cdot L_2$ è regolare.

  Nota: i due L_2 che compaiono nella definizione di L_3 non generano necessariamente la medesima stringa.

  Si può facilmente pensare che $L_3 \notin Reg$ e di provarlo con il pumping lemma. Ciò non è possibile, dunque viene il dubbio che L_3 sia regolare. Con una buona intuizione si arriva a $L_3 = \Sigma^* ab \Sigma^*$ (la dimostrazione, che prevede il caso $L_3 \subseteq \Sigma^* ab \Sigma^*$ e il caso $\supseteq$, non è necessaria), che è regolare. L'enunciato è vero.

  (In questi casi bisogna avere una buona intuizione. Il linguaggio, infatti, fornisce regole per costruire delle stringhe, talvolta ambigue e non univoche, quindi non bisogna fare completo affidamento sulla struttura che ne viene presentata.)

- L_1 e $L_2 \notin Reg$ allora $L_1 \cup L_2$ è sempre non regolare.

  Falso. Per dimostrarlo è sufficiente prendere un qualsiasi $L_1 \notin Reg$ e $L_2 = \bar{L}_1 \notin Reg$ (che è sempre vero, altrimenti si avrebbe $\bar{L}_1 \in Reg \Rightarrow L_1 \in Reg$ per chiusura rispetto alla complementazione[1]), quindi $L_1 \cup L_2 = L_1 \cup \bar{L}_1 = \Sigma^* \in Reg$.

  A lezione è stato fatto il seguente esempio:

  $$L_1 = \{a^n b^m |\ n \neq m,\ n, m \geq 1\} \notin Reg$$
  $$L_2 = \{a^n b^n |\ n \geq 1\} \notin Reg$$
  $$L_1 \cup L_2 = \{a^k b^h |\ k, h \geq 1\} \in Reg.$$

  Nota: a volte possono tornare utili gli operatori complementazione e reverse in quanto binari, come in questo caso[2].

- $L_1 \notin Reg$, $L_2 \in Reg$, allora $L_1 \cap L_2 \in Reg$.

  Falso, basta prendere un qualsiasi L_1 non regolare e $L_2 = \Sigma^*$: si ha $L_1 \cap L_2 = L_1 \cap \Sigma^* = L_1 \notin Reg$.

- $L = L_1 \cup L_2 \in Reg$ allora L_1 e L_2 sono regolari.

  Falso. La chiusura della classe Reg rispetto all'unione garantisce solo il viceversa! Siano per esempio L_1 non regolare e $L_2 = \Sigma^*$, la loro unione è regolare nonostante loro non lo siano.

- $L_1 \in CFL$ allora $\forall L_1' \subseteq L_1$ si ha $L_1' \in CFL$.

  Falso. Basta prendere $L_1 = \Sigma^*$ che è CFL perché regolare. Tutti i linguaggi, anche quelli $\notin CFL$ sono sottoinsiemi di Σ^*.

  Consiglio: quando si devono dimostrare proprietà che coinvolgono linguaggi di classi che contengono Reg o CFL, si rivela utile trovare esempi degeneri che coinvolgono proprio linguaggi Reg o CFL.

- $L_1 \in CFL$ e $L_2 \in Reg$ allora $L_1 \cap L_2$ può essere regolare.

  Vero. Prendendo $L_2 = \varnothing$ si ha $L_1 \cap L_2 = \varnothing$, regolare per ogni L_1.

  Seguendo il consiglio precedente, si può anche prendere un qualsiasi $L_1 \in Reg$: per la chiusura rispetto alla complementazione di Reg si ha $L_1 \cap L_2 \in Reg$.

[1] Abbiamo appena dimostrato che la *classe dei linguaggi non regolari* è chiusa rispetto all'operazione di complementazione.

[2] Può capitare, per esempio, che sia più facile dimostrare che il reverse di un linguaggio non sia regolare. Dimostrato questo, il linguaggio di partenza non può essere regolare.

- $L_1 \in CFL$ e $L_2 \in Reg$ allora $L_1 \cap L_2$ può non essere regolare.

  Vero. Si può prendere L_1 strettamente CFL, cioè $L_1 \in CFL/Reg$ e $L_2 = \Sigma^*$. Si ha $L_1 \cap L_2 = L_1 \notin Reg$.

- $L \in CFL \Rightarrow L' = \{w|\ w \in L, |w| = \text{pari}\} \in CFL$.

  Vero. Sia $L_1 = \{w|\ w \in \Sigma^*, |w| = \text{pari}\} \in Reg$. Si ha $L \cap L_1 = L' \in CFL$ perché intersezione di un CFL e un regolare.

- $L_1 \in CFL$, $L_2 \notin CFL$ allora $L_1 \cap L_2$ può essere $\notin CFL$.

  Vero, per dimostrarlo è sufficiente prendere un qualsiasi $L_2 \notin CFL$ e $L_1 = \Sigma^*$.

- $L_1 \in CFL$, L_2 finito, allora l'intersezione tra i due è CFL.

  Vero, l'intersezione di un linguaggio qualsiasi con un linguaggio finito dà un linguaggio finito, che è CFL.

- Sia $w \in \Sigma^*$,

  $$\pi(w) = \{u|\ u \text{ è una permutazione qualsiasi di } w\}.$$

  Si ha

  $$\pi(L) = \{u|\ \exists w \in L : u \in \pi(w)\}.$$

  Se $L \in Reg$ allora $\pi(L) \in CFL$.

  Dipende dall'alfabeto, più in particolare dalla sua cardinalità, che nell'enunciato non è specificata:

  - se $\Sigma = \{a, b\}$ allora sia, per esempio, $L_1 = L((\mathbf{ab})^*) \in Reg$. Si ha

    $$\pi(L_1) = \{w|\ \#_a(w) = \#_b(w)\} \in CFL.$$

  - se $\Sigma = \{a, b, c\}$, invece, con $L_2 = L((\mathbf{abc})^*)$ sempre regolare, si ha

    $$\pi(L) = \{w|\ \#_a(w) = \#_b(w) = \#_c(w)\} \notin CFL$$

    (ma ricorsivo perché è possibile scrivere un algoritmo che lo calcoli).

- $L = L^R$ allora $L \in Reg$.

  Falso. Un esempio è il linguaggio $L = \{a^n b a^n|\ n \geq 1\} \notin Reg$ che soddisfa $L = L^R$, oppure L_{pal}, il linguaggio delle stringhe palindrome $\in CFL$.

  Si noti che affinché un tale L appartenga a una classe non è necessario che tale classe sia chiusa rispetto all'operatore reverse.

- Sia $\Lambda(w) = w \cdot w^R$, e $\Lambda(L) = \{w \mid w = \Lambda(x), x \in L\}$. La classe dei linguaggi regolari è chiusa rispetto all'operatore Λ.

  Falso, lo si può dimostrare velocemente con il pumping lemma.

- La classe dei linguaggi finiti è chiusa rispetto all'applicazione di $\Lambda(\cdot)$.

  Vero, perché $|\Lambda(L)| = |L|$ per ogni L finito.

- Se $\bar{L}$ è finito, allora $\Lambda(L)$ è regolare.

  Falso. Se $\bar{L}$ è finito significa che L è infinito, quindi almeno regolare.

  Si noti che in generale le informazioni 'linguaggio infinito' e 'linguaggio complemento di un finito' sono diverse: L finito $\Rightarrow$ $\bar{L}$ infinito ma non vale il viceversa.

- $\Lambda(\bar{L}) = \bar{\Lambda}(L)$ per ogni L finito.

  Falso. Sia per esempio $L = \{a, b, \varepsilon\}$. Si verifica velocemente che

  $$\Lambda(\bar{L}) = \{ww^R \mid w \in \{a, b\}^n, n > 1\}$$

  mentre

  $$\bar{\Lambda}(L) = \{w | w \neq aa, w \neq bb, w \neq \varepsilon\}.$$

- Sia h un omomorfismo. $h(L) \in Reg \Rightarrow L \in Reg$.

  Falso. Sappiamo che $L \in Reg \Rightarrow h(L) \in Reg$ ma i teoremi di chiusura non garantiscono il viceversa. Un controesempio che avvalora quanto detto: $L = \{a^n b^n | \ n \geq 1\} \notin Reg$ e $h(a) = a$, $h(b) = a$. Allora $h(L) = \{a^m | \ m \text{ pari}\} \in Reg$.

- La classe REC è chiusa rispetto all'intersezione.

  Vero. Siano $L_1 = L(M_1)$ e $L_2 = L(M_2) \in REC$, allora $\exists M$ mdT tale che $L(M) = L_1 \cap L_2$: è sufficiente che tale macchina simuli M_1 e M_2 contemporaneamente sul medesimo input. Essa deve accettare se e solo se entrambe accettano, altrimenti no.

- La classe REC è chiusa rispetto alla complementazione.

  Vero. Sia $L = L(M) \in REC$. Una macchina che riconosca $\bar{L}$ è ottenuta da M invertendone l'output.

- La classe REC è chiusa rispetto all'unione.

  Vero. Si può mostrare una mdT che accetti l'unione numerabile di linguaggi ricorsivi a partire dalle macchine che accettano i rispettivi

linguaggi uniti, ma è sufficiente notare che per De Morgan (o equivalenza di σ e δ algebre) la chiusura rispetto all'unione è conseguenza della chiusura rispetto a intersezione e complementazione.

- La classe RE è chiusa rispetto all'intersezione.

  Vero. La mdT che accetta l'intersezione numerabile di linguaggi ricorsivamente enumerabili si può costruire con la medesima strategia utilizzata per costruire la mdT usata per dimostrare la chiusura dei REC rispetto all'intersezione.

- La classe RE è chiusa rispetto alla complementazione.

  Falso. Per un teorema studiato, se un $L \in RE$ è tale che $\bar{L} \in RE$ allora $L \in REC$. Se $L \in RE/REC$ allora si ha necessariamente $\bar{L} \notin RE$.

- La classe RE è chiusa rispetto all'unione.

  Vero. Si noti che aver dimostrato che vale la chiusura rispetto all'intersezione e non vale quella rispetto alla complementazione, non permette di concludere nulla circa l'unione. Per dimostrare che $L_1, L_2 \in RE \Rightarrow L_1 \cup L_2 \in RE$ è necessario esibire una macchina di Turing opportuna. Sia M tale macchina, siano M_1 e M_2 le macchine che accettano rispettivamente L_1 e L_2. La macchina M accetta se almeno una delle due accetta.

- $L_1 \in REC$ e $L_2 \in RE$ allora la loro intersezione è sempre ricorsiva.

  Falso, basta fornire un esempio. Sia $L_1 = \Sigma^* \in REC$ e $L_2 = L_u \in RE/REC$, allora $L_1 \cap L_2 = L_u$ ricorsivamente enumerabile ma non ricorsivo.

- Siano $L_1 \in REC$ e $L_2 \in RE$, allora $L_1 \cap L_2$ non è ricorsiva.

  Falso. Con $L_1 = L_2 = \varnothing$ si ha $L_1 \cap L_2 = \varnothing \in REC$.

- L_1 e $L_2 \in REC$, allora $L_1 \cdot L_2 \in REC$.

  Vero: la classe dei linguaggi ricorsivi è chiusa rispetto alla concatenazione.

  Per dimostrarlo è sufficiente esibire una mdT M che accetti L_1, accettato da M_1, concatenato a L_2, accettato da M_2. Per costruire M ci serviamo delle macchine M_1 e M_2. Sia $w = a_1 \dots a_n \in \Sigma^n$ l'input. Scegliamo non deterministicamente $i \in [1, n]$ tale che $w = xy$ con $x = a_1 \dots a_i$ e $y = a_{i+1} \dots a_n$. Diamo x in input a M_1 e y in input

a M_2. M accetta se e solo se entrambe accettano[3]. Notare che la macchina costruita è una NTM, poiché la scelta del *taglio* della stringa in input è operata in modo che, se possibile, le macchine accettino.

Per un teorema, una NTM ha la stessa potenza computazionale di un computer, quindi un modo alternativo di dimostrare la chiusura dei ricorsivi rispetto alla concatenazione è fornire un algoritmo. L'algoritmo è semplice:

Input: w

`for each` $i \in [1, |w|]$:

{ si generi $w = xy$ come prima

`if` M_1 accetta x `AND` M_2 accetta y `then` M accetta

}

Se dopo aver testato tutti i possibili tagli la macchina non ha ancora accettato, allora non accetta.

Nota: una prima risposta a tale domanda potrebbe essere 'mettere in serie le mdT così la seconda computa sull'input residuo lasciato dalla prima' ma ciò è errato in quanto le mdT non leggono l'input da sinistra a destra come i comuni automi a stati finiti.

0.1 Chiusure

Può essere utile, per questi esercizi, ricordare le proprietà di chiusura. Nota: sono omesse le relazioni tra linguaggi di classi diverse.

[3]Per costruire la AND degli output delle mdT si possono (o si dovrebbero) combinare gli output in tutti i modi possibili e opportunamente trattati.

$A, B \in$	finiti	Reg	CFL	REC	RE
$\bar{A}$		✓		✓	
$A \cup B$	✓	✓	✓	✓	✓
$A \cap B$	✓	✓		✓	✓
A^R	✓	✓	✓	✓	✓
$A \cdot B$	✓	✓	✓	✓	✓
A^*		✓	✓	✓	✓
A^+		✓	✓	✓	✓
$h(A)$	✓	✓	✓	✓	✓
A/B	✓	✓		✓	