Indice

1 Operazioni e chiusura dei CF (continua)

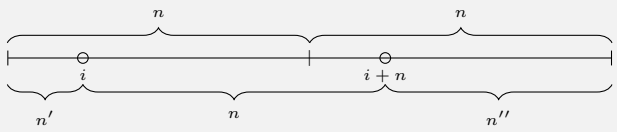
Esibiamo ora un controesempio che mostra che il complemento non è chiuso rispetto ai CF. Prendiamo il linguaggio

$$L = \{x \in \{a, b\}^* \mid \forall w : x \neq ww\} \in \text{CF}$$

il complemento di questo è

$$L^C = \{ww \mid w \in \{a, b\}^*\}$$

che come abbiamo già mostrato in passato non è context free.
Per completezza mostriamo anche che L sia effettivamente CF costruendo un automa a pila che lo riconosce. Questo – molto brevemente – scommette sulla posizione dei due caratteri diversi.



con $n' + n'' = n$ e $w_i \neq w_{i+n}$. Più precisamete l'automa segue le seguenti fasi

1. sulla pila viene contato n'.
2. si sceglie nondeterministicamente un terminale σ e si iniziano a togliere i n' simboli accumulati finché non si svuota la pila.
3. a questo punto si ricomincia a contare sulla pila fino a che non si sceglie un secondo terminale ρ che deve essere differente.
4. si ricomincia a togliere i simboli n'' dalla pila consumando l'input
5. deve valere che alla fine dell'input sia vuota anche la pila e viceversa, affinché $n' + n'' = n$

Lemma 1. *I DCFL sono chiusi rispetto al complemento.*

Uno dei problemi con gli automi a pila è che questi possono non terminare. Infatti utillizzando le ε-mosse l'automa può continuare a far oscillare la pila

o far crescere la pila all'infinito. Visto però che gli stati e i simboli della pila sono finiti, data una configurazione di superficie (*surface configuration*, o lo stato e il simbolo in cima alla pila) capire se ci sarà un ciclo infinito è decidibile, e consiste banalmente nel controllare se lo stesso stato e simbolo di pila si ripetono senza consumare input.

Inoltre visto che sono ammesse le ε-mosse, in un automa che accetta per stato finale, può accadere che una volta che l'input è finito l'automa continui a fare mosse, e addirittura può finire in un ciclo infinito. In ogni caso se c'è almeno uno stato finale tra quelli che visita dopo la fine dell'input, allora l'input è accettato.

Lemma 2. *Supponiamo che*

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$$

sia un automa a pila deterministico. Costruiamo

$$M' = \langle Q', \Sigma, \Gamma', \delta', q_0', X_0, F' \rangle$$

tale che

$$L(M) = L(M')$$

ed M' scandisce sempre l'intero input.

Dimostrazione. Definiamo l'automa M' tale che

- $Q' = Q \cup \{q_0', d, f\}$, con d detto *stato trappola*
- $\Gamma' = \Gamma \cup \{X_0\}$
- $F' = F \cup \{f\}$

e definiamo le mosse che fa l'automa

- $\delta'(q_0', \varepsilon, X_0) = (q_0, Z_0 X_0)$, cioè – come in altre costruzioni – "infiliamo" in fondo alla pila un nostro simbolo. Questo è necessario perché la prossima mossa potrebbe non essere definita perché si è svuotata la pila.
- mostriamo ora alcune regole particolari che portano allo stato trappola

– se in una certa configurazione di superficie dell'automa M non esiste una prossima mossa definita, cioè se

$$\delta(q, a, X) = \delta(q, \varepsilon, X) = \varnothing, \qquad q \in Q, a \in \Sigma, X \in \Gamma$$

allora nell'automa M' finisco nello stato trappola

$$\delta'(q, a, X) = (d, X)$$

– se nell'automa M la pila si è svuotata e non potrei pià fare mosse nell'automa M' finisco nello stato trappola

$$\delta'(q, a, X_0) = (d, X_0), \qquad q \in Q', a \in \Sigma$$

– nello stato trappola consumo l'input e rimango nello stato trappola

$$\delta'(d, a, X) = (d, X), \qquad a \in \Sigma, X \in \Gamma'$$

– se da (q, X), con $q \in Q$ e $X \in \Gamma$ è possibile un loop di ε-mosse

$$\delta'(q, \varepsilon, X) = \begin{cases} (d, X) & \text{se il loop non visita stati finali} \\ (f, X) & \text{altrimenti} \end{cases}$$

– se sono nello stato finale f e posso ancora leggere input, allora non sono alla fine quindi mi sposto nello stato trappola

$$\delta'(f, a, X) = (d, X), \qquad a \in \Sigma, X \in \Gamma'$$

- in tutti gli altri casi $\delta'(q, a, X) = \delta(q, a, X)$ con $q \in Q, a \in \Sigma \cup \{\varepsilon\}, X \in \Gamma$

■

Automa per il complemento. Sia

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$$

un DPDA che accetta L e scandisce sempre l'intero input. Costruiamo l'automa per il complemento

$$M' = \langle Q', \Sigma, \Gamma, \delta', q_0', Z_0, F' \rangle$$

con

- $Q' = Q \times \{y, n, A\}$, dove y, n, A servono a ricordare se nell'ultima sequenza di ε-mosse ho visto uno stato finale, rispettivamente:
  - y indica che nell'attuale stato della sequenza di ε-mosse è stato visitato uno stato finale.
  - n indica che nell'attuale stato della sequenza di ε-mosse non è stato visitato uno stato finale.
  - A indica che non ho visitato uno stato finale e non posso fare alcuna mossa.
- $F' = Q \times \{A\}$, cioè tutti gli stati in cui non posso procedere e non ho visitato uno stato finale

Definiamo ora δ' come

- se $\delta(q, \varepsilon, X) = (p, \gamma)$ allora
  - $\delta'([q, y], \varepsilon, X) = ([p, y], \gamma)$, quindi se in passato ho visitato uno stato finale, continuo a ricordare che ho visitato uno stato finale.
  - se non ho ancora visto uno stato finale, e il prossimo lo è, cambio n a y
$$\delta'([q, n], \varepsilon, X) = \begin{cases} ([p, n], \gamma) & \text{se } p \notin F \\ ([p, y], \gamma) & \text{se } p \in F \end{cases}$$
- se $\delta(q, a, X) = (p, \gamma)$ con $a \in \Sigma$, allora
  - se ero in uno stato y e consumo dell'input e finisco in uno stato non finale, cambio y a n
$$\delta'([q, y], a, X) = \begin{cases} ([p, n], \gamma) & \text{se } p \notin F \\ ([p, y], \gamma) & \text{se } p \in F \end{cases}$$
  - alternativametne se ero in uno stato n spezzo la mossa in due parti
$$\delta'([q, n], \varepsilon, X) = ([q, A], X)$$
$$\delta'([q, A], a, X) = \begin{cases} ([p, n], \gamma) & \text{se } p \notin F \\ ([p, y], \gamma) & \text{se } p \in F \end{cases}$$

cioè prima di consumare dell'input assumo di aver finito la stringa senza esser passato per stati finali nella sequenza di ε-mosse. Se c'è ancora in input trasformo lo A in y o n.

Infine definiamo

$$q_0' = \begin{cases} [q_0, n] & \text{se } q_0 \notin F \\ [q_0, y] & \text{se } q_0 \in F \end{cases}$$

■

Dalla costruzione di sopra si vede che le ε-mosse sono molto fastidiose. Come abbiamo visto attraverso la trasformazione in GNF nel caso di PDA si può fare a meno delle ε-mosse a patto di sacrificare la parola vuota. Lo stesso non vale nel caso di DPDA, e senza ε-mosse otteniamo un modello meno potente.

Vediamo ora altre operazioni

Operazione	CFL?	DCFL?
Unione	Sì	No
Intersezione	No	No
Intersezione con un regolare	Sì	Sì
Complemento	No	Sì
Prodotto	Sì	No
Chiusura di Kleene	Sì	No
Morfismo	Sì	No
Sostituzione	Sì	No
Reversal	Sì	No
Shuffle	No	No

Tabella 1: Chisura delle operazioni

Prodotto Non deterministicamente quando un automa arriva in fondo faccio partire l'automa successivo. Questo si può fare anche in termini di grammatiche, date $G' = \langle V', \Sigma, P', S' \rangle$ e $G'' = \langle V'', \Sigma, P'', S'' \rangle$ con $V' \cap V'' = \varnothing$ creiamo

$$G = \langle V' \cup V'' \cup \{S\}, \Sigma, P' \cup P'' \cup \{S \to S'S''\}, S \rangle$$

Per il caso deterministico utilizziamo L_0 che abbiamo visto dopo e definiamo

$$L' = \{\varepsilon, d\} \cdot L_0 = \{d^s a^i b^j c^k \mid 0 \leq s \leq 2, \begin{cases} s = 0 & i = j \\ s = 2 & j = k \\ s = 1 & i = j \vee j = k \end{cases} \}$$

Per mostrare che è ambiguo prendiamo

$$L' \cap da^* b^* c^* = \{da^i b^j c^k \mid i = j \vee j = k\} \notin \text{DCFL}$$

con $da^*b^*c^*$ regolare, visto che i deterministici sono chiusi rispetto all'intersezione con regolari, L' non può essere deterministico. Inoltre $\{\varepsilon, d\}$ è finito, quindi i linguaggi deterministici non sono chiusi rispetto al prodotto a sinistra con linguaggi finiti. Mostriamo però che se $L \in$ DCFL e $R \in$ Reg il prodotto $L \cdot R \in$ DCFL. Questa costruzione è simile al prodotto per gli automi a stati finiti.

Chiusura di Kleene – star Molto simile al prodotto. Data la grammatica $G' = \langle V', \Sigma, P', S' \rangle$, costruiamo

$$G = \langle V' \cup \{S\}, \Sigma, P' \cup \{S \to \varepsilon, S \to S'S\}, S \rangle$$

Nel caso dei deterministici prendiamo ancora L_0 e analizziamo

$$L_0^* \cap da^* b^* c^* = d(L_1 \cup L_2)$$

infatti questo ha stringhe della forma

$$da^i b^j c^k$$

tali che

- dL_2, per cui $j = k$
- o il prodotto $d \in L_2$ per $a^i b^j c^j \in L_1$, per cui $i = j$

ma questo abbiamo visto che non è deterministico, quindi L_0^* non è chiuso rispetto alla star.

Morfismo Per ogni terminale $a \in \Sigma$, lo sostituisco con una stringa $w \in \Delta^*$ utilizzando una funzione

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

Per questo basta sostituire i terminali in ogni produzione. Nel caso dei deterministici prendiamo

$$L' = \{a^i b^j c^k \mid i = j\}$$
$$L'' = \{a^i b^j c^k \mid j = k\}$$

e sappiamo che $L', L'' \in$ DCFL, e che $L' \cup L'' \notin$ DCFL. Mentre vale che $L_0 = L' \cup dL'' \in$ DCFL. Prendiamo il morfismo

$$h(\sigma) = \begin{cases} \sigma & \text{se } \sigma \neq d \\ \varepsilon & \text{altrimenti} \end{cases}$$

vale che $h(L_0) = L' \cup L'' \notin$ DCFL, quindi i determinisici non sono chiusi rispetto al morfismo.

Sostituzione Ad ogni terminale associamo un linguaggio, utilizzando una funzione

$$s : \Sigma \to 2^{\Delta^*}$$

Se $L \in$ CFL e $\forall a \in \Sigma \mid s(a) \in$ CFL, allora $s(L) \in$ CFL. Molto ad alto livello ad ogni terminale corrisponde una grammatica, nelle produzioni sostituiamo ai terminali l'assioma della grammatica. Visto che il morfismo è un caso particolare della sostituzione, i determinisitici non sono chiusi rispetto alla sostituzione.