Automi e Linguaggi Formali
Università degli Studi di Padova
Aprile 2024

Homework 5 - Esercizi in preparazione del primo compitino

Gabriel Rovesti

1. Dati i linguaggi A e B, l'*interleaving* dei due linguaggi è definito come: $\{w \mid W = a_1b_1...a_kb_k \mid a_1...a_k \in A, b_1...b_k \in B, \forall a_i, b_i \in \Sigma^*\}$
   Dimostra che la classe dei linguaggi regolari è chiusa per l'operazione di interleaving.

   Per dimostrare che la classe dei linguaggi regolari è chiusa per l'operazione di interleaving, dobbiamo dimostrare che l'interleaving di due linguaggi regolari produce un linguaggio regolare. Per fare ciò, possiamo seguire un approccio di dimostrazione per costruzione tramite un automa a stati finiti (DFA) che riconosce l'interleaving di due linguaggi regolari così definiti.

   Supponiamo di avere due linguaggi regolari A e B.

   Sia $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ un automa che riconosce il linguaggio A, e sia $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ un automa che riconosce il linguaggio B.

   Costruiremo un nuovo automa M che riconosce l'interleaving. L'idea è di utilizzare un NFA che tiene traccia delle possibili transizioni simultanee tra M_A e M_B.

   Definiamo M come segue:

   (a) **Stati**: Gli stati di M saranno coppie ordinate di stati, uno da M_A e uno da M_B. Formalmente, $Q = Q_A \times Q_B$.

   (b) **Alfabeto**: L'alfabeto di M è lo stesso di M_A e M_B, quindi Σ.

   (c) **Funzione di transizione**: La funzione di transizione di M è definita come segue:

   $$\delta((q_A, q_B), a) = (\delta_A(q_A, a), q_B) \cup (q_A, \delta_B(q_B, a))$$

   Questa definizione riflette il fatto che l'automa può leggere un simbolo a contemporaneamente da entrambi i linguaggi A e B. L'automa passa allo stato successivo in M_A o M_B (o entrambi) a seconda di dove si trova il simbolo.

(d) **Stato iniziale**: Lo stato iniziale di M sarà la coppia degli stati iniziali di M_A e M_B, cioè $q_0 = (q_{0A}, q_{0B})$.

(e) **Stati finali**: Uno stato q in M è finale se e solo se entrambi gli stati corrispondenti in M_A e M_B sono finali. Formalmente, $F = \{(q_A, q_B) \mid q_A \in F_A \wedge q_B \in F_B\}$.

Ora dobbiamo dimostrare che M riconosce l'interleaving $A \,||\, B$. Ci sono due parti principali:

(a) Se $w \in A \,||\, B$, allora M accetta w.

(b) Se M accetta w, allora $w \in A \,||\, B$.

La dimostrazione di entrambe queste parti è una diretta conseguenza della definizione di M e del fatto che M_A e M_B riconoscono rispettivamente i linguaggi A e B.

Quindi, poiché abbiamo costruito un DFA M che riconosce l'interleaving di due linguaggi regolari A e B, possiamo concludere che la classe dei linguaggi regolari è chiusa per l'operazione di interleaving.

2. Se A e B sono due linguaggi, definiamo A $\triangle$ B $= \{xy \mid x \in A, y \in B, |x| = |y|\}$. Si dimostri che se A e B sono linguaggi regolari, allora $A \triangle B$ è un linguaggio context-free.

   Per dimostrare che se A e B sono linguaggi regolari, allora $A \triangle B$ è un linguaggio context-free, possiamo costruire una grammatica context-free che genera $A \triangle B$.

   Siano A e B linguaggi regolari. Poiché la classe dei linguaggi regolari è chiusa sotto unione, possiamo assumere senza perdita di generalità che A e B non contengano la stringa vuota ϵ.

   Sia $G_A = (N_A, \Sigma, P_A, S_A)$ una grammatica regolare che genera A, e $G_B = (N_B, \Sigma, P_B, S_B)$ una grammatica regolare che genera B.

   Costruiamo una grammatica context-free $G = (N, \Sigma, P, S)$ che genera $A \triangle B$ nel seguente modo:

   $N = N_A \cup N_B \cup \{S, X, Y\}$, dove S è il nuovo simbolo iniziale, e X e Y sono nuovi simboli non terminali.

   P contiene tutte le produzioni di P_A e P_B, più le seguenti produzioni:

   (a) $S \to XY$

   (b) $X \to aX \mid a$, per ogni $a \in \Sigma$ tale che esiste una produzione $S_A \to a\alpha$ in P_A

(c) $Y \to bY \mid b$, per ogni $b \in \Sigma$ tale che esiste una produzione $S_B \to b\beta$ in P_B

3. Un all-NFA M è una tupla $(Q, \Sigma, \delta, q_0, F)$ che accetta $x \in \Sigma^*$ se *ogni* possibile stato che M ottiene dopo aver letto l'input x è uno stato in F. Si noti che, rispetto ad un NFA ordinario, accetta una stringa se *qualche* stato tra quelli possibili è uno stato di accettazione. Si provi che gli all-NFA sono chiusi rispetto alla classe dei linguaggi regolari.

   Per dimostrare che gli all-NFA sono chiusi rispetto alla classe dei linguaggi regolari, dobbiamo mostrare che per ogni linguaggio regolare L, esiste un all-NFA che riconosce L.

   La dimostrazione procede costruendo, per un dato NFA M che riconosce L, un all-NFA M' che riconosce esattamente L.

   Sia $M = (Q, \Sigma, \delta, q_0, F)$ un NFA che riconosce il linguaggio regolare L. Costruiamo l'all-NFA $M' = (Q', \Sigma, \delta', q_0', F')$ nel seguente modo:

   (a) $Q' = 2^Q \cup \{q_{\text{sink}}\}$, dove 2^Q è l'insieme delle parti di Q e q_{sink} è un nuovo stato.

   (b) $q_0' = \{q_0\}$

   (c) $F' = \{S \subseteq Q \mid S \subseteq F\}$, cioè gli stati di accettazione di M' sono tutti e soli gli insiemi di stati che sono sottoinsiemi degli stati di accettazione di M.

   (d) La funzione di transizione δ' è definita come segue:
   - Per ogni $S \subseteq Q$ e $a \in \Sigma$, $\delta'(S, a) = \bigcup\{\delta(q, a) \mid q \in S\} \cup \{q_{\text{sink}}\}$
   - Per ogni $S \subseteq Q$, $\delta'(S, \epsilon) = S$
   - Per ogni $a \in \Sigma$, $\delta'(q_{\text{sink}}, a) = \{q_{\text{sink}}\}$

   Intuitivamente, M' tiene traccia dell'insieme degli stati raggiungibili in M dopo aver letto una certa porzione dell'input. Se durante una transizione si raggiunge l'insieme vuoto, M' transita nello stato q_{sink}, uno stato di non accettazione da cui non può più uscire. M' accetta una stringa x se e solo se tutti gli stati raggiungibili dopo aver letto x sono stati di accettazione in M.

   Dimostriamo che $L(M') = L(M)$:

   - ($\subseteq$) Supponiamo che $x \in L(M')$. Allora M', dopo aver letto x, si trova in un insieme $S \subseteq F$ di stati di accettazione di M. Quindi, per ogni $q \in S$, q è uno stato raggiungibile in M dopo aver letto x. Poiché $S \subseteq F$, segue che almeno uno di questi stati è uno stato di accettazione di M. Quindi, $x \in L(M)$.

- ($\supseteq$) Supponiamo che $x \in L(M)$. Allora esiste uno stato $q \in F$ raggiungibile in M dopo aver letto x. Consideriamo l'insieme S di stati raggiungibili in M' dopo aver letto x. Poiché M' simula M, $q \in S$. Inoltre, per la definizione di F', $S \subseteq F$. Quindi, M' accetta x.

Abbiamo dimostrato che $L(M') = L(M)$, e poiché M' è un all-NFA per costruzione, segue che gli all-NFA sono chiusi rispetto alla classe dei linguaggi regolari.

4. Se A è un qualsiasi linguaggio, sia $A_{1/2-}$ l'insieme di tutte le prime metà di stringa in A tali che: $A_{1/2-} = \{x \mid$ per qualche $y, |x| = |y|$ e $xy \in A\}$. Si dimostri che, se A è regolare, allora anche $A_{1/2-}$ lo è.

   Per dimostrare che se A è un linguaggio regolare, allora anche $A_{1/2-}$ è regolare, possiamo costruire un automa a stati finiti non deterministico (NFA) che riconosce $A_{1/2-}$ a partire da un NFA che riconosce A.

   Sia $M = (Q, \Sigma, \delta, q_0, F)$ un NFA che riconosce il linguaggio regolare A. Costruiamo un NFA $M' = (Q', \Sigma, \delta', q_0', F')$ che riconosce $A_{1/2-}$ nel seguente modo:

   (a) $Q' = Q \times \{0, 1\}$, dove ogni stato di M' è una coppia (p, b) con $p \in Q$ e $b \in \{0, 1\}$. L'idea è che $b = 0$ indica che M' non ha ancora raggiunto la metà della stringa, mentre $b = 1$ indica che è stata raggiunta la metà della stringa.

   (b) $q_0' = (q_0, 0)$, lo stato iniziale di M' è la coppia formata dallo stato iniziale di M e il flag 0.

   (c) $F' = \{(p, 1) \mid p \in F\}$, gli stati finali di M' sono tutte le coppie in cui lo stato di M è uno stato finale e il flag è 1.

   (d) La funzione di transizione δ' è definita come segue:

   - $\delta'((p, 0), a) = \{(q, 0) \mid q \in \delta(p, a)\}$, se il flag è 0, M' simula M senza cambiare il flag.
   - $\delta'((p, 0), a) = \{(q, 1) \mid q \in \delta(p, a)\}$, se il flag è 0 e la transizione avviene su un simbolo a, M' simula M e imposta il flag a 1, indicando che è stata raggiunta la metà della stringa.
   - $\delta'((p, 1), a) = \{(q, 1) \mid q \in \delta(p, a)\}$, se il flag è 1, M' simula M senza cambiare il flag.

   Intuitivamente, M' simula M fino a raggiungere la metà della stringa di input, dopodiché continua a simulare M ma con il flag 1 attivo. M'

accetta una stringa x se e solo se x è la prima metà di una stringa $xy \in A$.

Dimostriamo che $L(M') = A_{1/2-}$:

- ($\subseteq$) Supponiamo che $x \in L(M')$. Allora M', dopo aver letto x, si trova in uno stato $(p, 1)$ con $p \in F$. Quindi, c'è una stringa y tale che xy è accettata da M, e siccome M' ha raggiunto il flag 1 dopo aver letto x, segue che $|x| = |y|$. Quindi, $x \in A_{1/2-}$.
- ($\supseteq$) Supponiamo che $x \in A_{1/2-}$. Allora esiste una stringa y tale che $|x| = |y|$ e $xy \in A$. Poiché M riconosce A, dopo aver letto xy, M raggiunge uno stato finale $p \in F$. Inoltre, poiché M' simula M fino a raggiungere la metà della stringa, dopo aver letto x, M' raggiunge lo stato $(p, 1)$, che è uno stato finale di M'. Quindi, $x \in L(M')$.

Abbiamo dimostrato che $L(M') = A_{1/2-}$, e poiché M' è un NFA per costruzione, segue che $A_{1/2-}$ è un linguaggio regolare.

Questa dimostrazione stabilisce che se A è un linguaggio regolare, allora anche $A_{1/2-}$ è un linguaggio regolare, dimostrando così che la classe dei linguaggi regolari è chiusa rispetto all'operazione $A_{1/2-}$.

5. Sia $\Sigma = \{1, \#\}$ e sia $Y = \{w \mid w = x_1\#x_2\#...\#x_k$ per qualche $k \geq 0, \forall x_i \in 1^*, x_i \neq x_j$ per $i \neq j\}$. Si dimostri che Y non è regolare.

   Per dimostrare che il linguaggio $Y = \{w \mid w = x_1\#x_2\#...\#x_k$ per qualche $k \geq 0, \forall x_i \in 1^*, x_i \neq x_j$ per $i \neq j\}$ sull'alfabeto $\Sigma = \{1, \#\}$ non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari.

   Il pumping lemma afferma che se un linguaggio L è regolare, allora esiste una costante n (dipendente da L) tale che ogni stringa $w \in L$ con $|w| \geq n$ può essere scritta come $w = xyz$, soddisfacendo le seguenti condizioni:

   (a) $|xy| \leq n$

   (b) $|y| \neq \epsilon$

   (c) $xy^iz \in L$ per ogni $i \geq 0$

   Dimostreremo che il linguaggio Y non soddisfa questa proprietà, contraddicendo quindi il pumping lemma e concludendo che Y non è regolare.

Supponiamo, per assurdo, che Y sia regolare. Allora, per il pumping lemma, esiste una costante n tale che ogni stringa $w \in Y$ con $|w| \geq n$ può essere scritta come $w = xyz$ soddisfacendo le condizioni del pumping lemma.

Consideriamo la stringa $w = 1^n\#1^n\#1^n\#...\#1^n$, con n occorrenze di 1^n separate da $\#$. Chiaramente, $w \in Y$ e $|w| \geq n$.

Poiché w soddisfa le condizioni del pumping lemma, possiamo scrivere $w = xyz$ con $|xy| \leq n$, $|y| \geq 1$ e $xy^iz \in Y$ per ogni $i \geq 0$.

Siccome $|xy| \leq n$, la stringa xy può contenere al massimo una occorrenza completa di $1^n\#$. Questo implica che y deve contenere una parte di un'occorrenza di $1^n\#$, altrimenti xy^2z non apparterrebbe a Y (poiché violerebbe la condizione che le sottostringhe di 1^* separate da $\#$ devono essere diverse).

Tuttavia, se y contiene una parte di un'occorrenza di $1^n\#$, allora $xy^0z = xz \notin Y$, poiché xz non contiene alcuna occorrenza completa di $1^n\#$.

Abbiamo quindi trovato una stringa $w \in Y$ con $|w| \geq n$ tale che non può essere scritta come $w = xyz$ soddisfacendo le condizioni del pumping lemma. Questa contraddizione dimostra che Y non può essere un linguaggio regolare.

Quindi, abbiamo dimostrato che il linguaggio $Y = \{w \mid w = x_1\#x_2\#...\#x_k$ per qualche $k \geq 0, \forall x_i \in 1^*, x_i \neq x_j$ per $i \neq j\}$ sull'alfabeto $\Sigma = \{1, \#\}$ non è regolare.

6. Sia $\Sigma = \{0, 1, +, =\}$ e sia $ADD = \{x = y + z \mid x, y, z$ siano interi binari e x sia la somma di y e di $z\}$. Si dimostri che ADD non è regolare.

   Idea: ci deve essere solo un $-$ e un $+$ in qualsiasi stringa di ADD, quindi la stringa ripetuta deve essere in uno dei numeri binari. Ma naturalmente cambiare il termine della somma rende l'equazione sbagliata, e cambiare i numeri aggiunti cambia la loro somma, rendendola anch'essa sbagliata.

   Supponiamo per assurdo che ADD sia regolare e sia p la sua pumping length. Sia $s = 1^{p+1} = 10^p + 1^p$. È evidente che questa equazione è vera, quindi $s \in L$. Ora, $|s| \geq p$, quindi possiamo scrivere $s = xyz$ con $|xy| \leq p, |y| > 0$ e $xz \in L$.
   Poiché i primi p caratteri di s sono tutti 1 e $|xy| \leq p$, y deve essere composto solo da 1, quindi $xz = 1^{p-|y|} = 10^p + 1^p$. Poiché $|y| > 0$, xz ha una stringa diversa a sinistra del segno di uguale rispetto a s, ma per il resto è identico.

Poiché le espansioni binarie sono uniche, l'equazione data da xz non può essere vera (altrimenti entrambe le stringhe sarebbero espansioni binarie del numero a cui si valuta il loro lato destro comune). Quindi $xz \notin L$, dimostrando che L non è regolare.