Automi e Linguaggi Formali – Appello 9 Settembre 2024

Esercizio 1 (9 punti): L non è regolare

Teorema

Il linguaggio $L = \{x1^n \mid x \in \{0,1\}^* \text{ e } n \geq |x|\}$ non è regolare.

Dimostrazione

Procediamo per assurdo usando il **Pumping Lemma**.

Assunzione: Supponiamo che L sia regolare.

Applicazione del Pumping Lemma: Esiste $p > 0$ tale che ogni stringa $s \in L$ con $|s| \geq p$ può essere decomposta come $s = xyz$ con:

1. $|xy| \leq p$
2. $|y| > 0$
3. $xy^i z \in L$ per ogni $i \geq 0$

Scelta della stringa test: Consideriamo $s = 0^p 1^p$.

Verifica $s \in L$:

- $x = 0^p \in \{0,1\}^*$
- $n = p$, $|x| = p$
- Condizione: $n \geq |x| \Leftrightarrow p \geq p$ ✓

Inoltre $|s| = 2p \geq p$.

Decomposizione: Dato $|xy| \leq p$ e $s = 0^p 1^p$, la sottostringa xy è contenuta nella parte degli 0.

Esistono $j, k \geq 0$ con $j + k \leq p$ e $k > 0$ tali che:

- $x = 0^j$
- $y = 0^k$
- $z = 0^{p-j-k} 1^p$

Test di pumping (i = 0): $xy^0z = xz$ = 0^j 0^{p-j-k} 1^p = 0^{p-k} 1^p

Verifica appartenenza: Per 0^{p-k} 1^p ∈ L, deve valere la forma w1^m con m ≥ |w|.

L'unica decomposizione è w = 0^{p-k}, m = p.
Condizione: m ≥ |w| ⟺ p ≥ p-k ⟺ k ≥ 0 ✓

Test di pumping (i = 2): xy^2z = 0^{p+k} 1^p

Decomposizione: w = 0^{p+k}, m = p.
Condizione: m ≥ |w| ⟺ p ≥ p+k ⟺ 0 ≥ k

Ma k > 0 per il Pumping Lemma, quindi 0 ≥ k è falso.

Contraddizione: $xy^2z \notin L$, violando il Pumping Lemma.

Quindi L non è regolare. ■

Esercizio 2 (9 punti): Flip e linguaggi context-free

Teorema

La classe dei linguaggi context-free è chiusa rispetto all'operazione di flip.

Dimostrazione

Definizione: Data w ∈ {0,1}*, flip(w) si ottiene scambiando tutti gli 0 con 1 e viceversa. flip(L) = {flip(w) | w ∈ L}.

Costruzione: Dato un PDA P = $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ che riconosce L, costruiamo P' che riconosce flip(L).

PDA P': P' = $(Q, \Sigma, \Gamma, \delta', q_0, Z_0, F)$

dove δ' è definita da:

- Se δ(q, a, X) contiene (p, α), allora δ'(q, flip(a), X) contiene (p, α)
- Se δ(q, ε, X) contiene (p, α), allora δ'(q, ε, X) contiene (p, α)

Correttezza: P' simula P sostituendo ogni simbolo di input con il suo flip.

Lemma: w ∈ L(P) ⟺ flip(w) ∈ L(P')

Dimostrazione del lemma: Per induzione sulla lunghezza delle computazioni.

- Ogni transizione di P su input a corrisponde a una transizione di P' su flip(a)
- Le ε-transizioni rimangono identiche
- Gli stati di accettazione sono preservati

Quindi la classe CF è chiusa rispetto al flip. ■

Esercizio 3 (9 punti): Automa a coda equivalente a Turing Machine

Teorema

Gli automi a coda sono equivalenti alle Turing Machine.

Dimostrazione

Definizione automa a coda: Un automa a coda è simile a un PDA ma la coda sostituisce la pila:

- Solo accesso FIFO (First In, First Out)
- Operazioni: enqueue (push), dequeue (pull)

Direzione 1: Turing Machine → Automa a coda

Data una TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q\_accept, q\_reject)$, costruiamo un automa a coda A.

Idea: Simulare il nastro della TM usando la coda.

Codifica del nastro: Il contenuto del nastro $\ldots a_{-2}a_{-1}[a_0]a_1a_2\ldots$ (con testina su a_0) viene codificato nella coda come:

```
⊢ a₋₂ a₋₁ â₀ a₁ a₂ ... ⊣
```

dove $\hat{a}_0$ indica la posizione della testina.

Simulazione di una transizione δ(q, a) = (q', b, L):

1. Dequeue finché non trovi â (simbolo marcato)
2. Enqueue tutti i simboli dequeue-ati (eccetto â)
3. Enqueue il nuovo simbolo b
4. Marca il simbolo precedente (simulando movimento L)
5. Enqueue il resto

Gestione movimento finito: Ogni simulazione di passo richiede tempo lineare nella lunghezza del nastro.

Direzione 2: Automa a coda → Turing Machine

Dato un automa a coda A, costruiamo una TM M che simula la coda sul nastro.

Codifica della coda: Rappresentare la coda come stringa sul nastro:

```
⊢ head → ... → tail ⊣
```

Simulazione enqueue(x):

1. Vai alla fine della coda (cerca ⊣)
2. Scrivi x prima di ⊣
3. Torna all'inizio

Simulazione dequeue():

1. Leggi il primo elemento dopo ⊢
2. Shifta tutto verso sinistra
3. Torna all'inizio

Correttezza: Entrambe le simulazioni preservano la semantica computazionale.

Quindi automi a coda ≡ Turing Machine. ■

Esercizio 4 (9 punti): Circuito semi-Hamiltoniano

Definizione

Un circuito semi-Hamiltoniano in G è un ciclo che attraversa tutti i vertici di G, passando per ogni vertice non più di due volte, ma non tre volte o più.

Parte (a): Il problema è in NP

Certificato: Una sequenza di vertici $C = v_1, v_2, \ldots, v_k, v_1$.

Verifica (tempo polinomiale):

1. **Controllo ciclo:** $v_1 = v_{k+1}$ e $(v_i, v_{i+1}) \in E$ per ogni i

2. **Controllo copertura:** Per ogni $v \in V$, esiste almeno un i tale che $v_i = v$
3. **Controllo semi-Hamiltoniano:** Per ogni $v \in V$, v appare al massimo 2 volte in C

Tutti i controlli richiedono tempo $O(|V| + |E|)$.

Quindi il problema è in NP. ■

Parte (b): Il problema è NP-hard

Riduzione dal Circuito Hamiltoniano.

Costruzione: Dato $G = (V, E)$, costruiamo $G' = G$ (stesso grafo).

Correttezza della riduzione:

Direzione (⇒): Se G ha circuito Hamiltoniano H

- H attraversa ogni vertice esattamente una volta
- Quindi H è anche semi-Hamiltoniano (≤ 2 volte per vertice)
- G' = G ha circuito semi-Hamiltoniano

Direzione (⇐): Se G' ha circuito semi-Hamiltoniano C

- C deve attraversare tutti i vertici di G' = G
- Ogni vertice è attraversato ≤ 2 volte

Lemma chiave: Se esiste un circuito semi-Hamiltoniano, allora esiste un circuito Hamiltoniano.

Dimostrazione del lemma: Sia C un circuito semi-Hamiltoniano. Se C attraversa qualche vertice v esattamente 2 volte, possiamo "accorciare" C:

- Identifica le due occorrenze di v in C
- Rimuovi il segmento tra le due occorrenze
- Ottieni un circuito più corto che ancora copre tutti i vertici

Iterando questo processo, otteniamo un circuito Hamiltoniano.

Conclusione: Circuito Hamiltoniano $\leq_p$ Circuito semi-Hamiltoniano

Quindi il problema è NP-hard.

Combinando (a) e (b): **Circuito semi-Hamiltoniano è NP-completo.** ■