Automi e Linguaggi Formali - Esame del 23 Luglio 2024

Problema 1 (9 punti)

Considera il linguaggio L = {1^m 0^n | 5m ≤ 3n}. Dimostra che L non è regolare.

Dimostrazione per contraddizione usando il Pumping Lemma

Assunzione: Supponiamo per contraddizione che L sia regolare.

Applicazione del Pumping Lemma: Allora esiste una costante p > 0 (pumping length) tale che ogni stringa w ∈ L con |w| ≥ p può essere decomposta come w = xyz con:

1. |xy| ≤ p
2. |y| > 0
3. xy^i z ∈ L per ogni i ≥ 0

Scelta della stringa di test: Consideriamo w = 1^p 0^⌈5p/3⌉ ∈ L.

Verifichiamo che w ∈ L: dobbiamo avere 5m ≤ 3n, cioè 5p ≤ 3⌈5p/3⌉.
Poiché ⌈5p/3⌉ ≥ 5p/3, abbiamo 3⌈5p/3⌉ ≥ 5p, quindi w ∈ L.

Analisi della decomposizione: Poiché |xy| ≤ p e w inizia con p occorrenze di 1, la substring xy deve essere contenuta interamente nella parte dei '1'. Quindi:

- x = 1^a per qualche a ≥ 0
- y = 1^b per qualche b > 0 (da |y| > 0)
- z = 1^(p-a-b) 0^⌈5p/3⌉

Derivazione della contraddizione: Consideriamo xy^0z = xz = 1^(p-b) 0^⌈5p/3⌉.

Per essere in L, questa stringa deve soddisfare:
5(p-b) ≤ 3⌈5p/3⌉

Ma sappiamo che 3⌈5p/3⌉ ≥ 5p (come verificato sopra), quindi:
5(p-b) ≤ 5p
p-b ≤ p
-b ≤ 0
b ≥ 0

Questo è sempre vero. Dobbiamo essere più precisi.

Approccio corretto: Consideriamo w = 1^(3k) 0^(5k) per k sufficientemente grande.
Verifichiamo: 5(3k) = 15k ≤ 3(5k) = 15k ✓

Nella decomposizione xyz:

- y = 1^b con 1 ≤ b ≤ p
- Consideriamo xy^2z = 1^(3k+b) 0^(5k)

Per essere in L: 5(3k+b) ≤ 3(5k) = 15k
Quindi: 15k + 5b ≤ 15k, che implica 5b ≤ 0, cioè b ≤ 0.

Ma sappiamo che b > 0 dal pumping lemma. **Contraddizione!**

Quindi L non è regolare. □

Problema 2 (9 punti)

*Dimostra che se L ⊆ Σ è context-free e T è una traslitterazione, allora T(L) è context-free.**

Dimostrazione costruttiva

Definizione: Una traslitterazione T: Σ → Γ* mappa ogni simbolo a ∈ Σ in una stringa T(a) ∈ Γ*.

Estensione a stringhe: $T(a_1a_2...a_n) = T(a_1)T(a_2)...T(a_n)$

Costruzione della grammatica: Sia G = (V, Σ, R, S) una CFG per L. Costruiamo G' = (V, Γ, R', S) dove:

R' è ottenuta da R sostituendo ogni produzione A → α con A → T(α), dove:

- Se α = a ∈ Σ, allora T(α) = T(a)
- Se $\alpha = A_1A_2...A_k$ (variabili), allora $T(\alpha) = A_1A_2...A_k$
- Se $\alpha = a_1A_1a_2A_2...a_kA_ka_{k+1}$, allora $T(\alpha) = T(a_1)A_1T(a_2)A_2...T(a_k)A_kT(a_{k+1})$

Correttezza:

1. **T(L) ⊆ L(G'):** Se w ∈ T(L), allora w = T(u) per qualche u ∈ L. Poiché u ∈ L(G), esiste una derivazione S ⇒* u in G. La stessa sequenza di applicazioni di produzioni in G' produce S ⇒* T(u) = w.

2. **L(G') ⊆ T(L):** Se w ∈ L(G'), esiste una derivazione S ⇒* w in G'. Questa derivazione corrisponde a una derivazione S ⇒* u in G per qualche u ∈ Σ*, e w = T(u). Quindi w ∈ T(L).

Pertanto T(L) = L(G') è context-free. □

Problema 3 (9 punti)

Mostra che i 2-PDA sono equivalenti alle Turing Machine.

Dimostrazione di equivalenza

Dobbiamo dimostrare che 2-PDA ≡ TM, cioè 2-PDA ⊆ TM e TM ⊆ 2-PDA.

Parte 1: 2-PDA ⊆ TM Ogni 2-PDA può essere simulato da una TM:

- La TM usa il nastro per simulare entrambe le pile del 2-PDA
- Divide il nastro in tre sezioni: input, pila 1, pila 2
- Le operazioni push/pop sono simulate da movimenti sul nastro
- Il controllo finito del 2-PDA è implementato negli stati della TM

Parte 2: TM ⊆ 2-PDA (costruzione chiave) Ogni TM M può essere simulata da un 2-PDA P:

Rappresentazione del nastro:

- Pila 1: contiene la parte sinistra del nastro (dalla posizione della testina verso sinistra)
- Pila 2: contiene la parte destra del nastro (dalla posizione della testina verso destra)
- Il simbolo in cima a pila 2 è quello sotto la testina

Simulazione delle transizioni: Per una transizione δ(q, a) = (q', b, D):

1. **Movimento a destra (D = R):**
   - Pop a da pila 2
   - Push b su pila 1
   - Se pila 2 è vuota, push ⊔ (blank)
2. **Movimento a sinistra (D = L):**
   - Pop a da pila 2
   - Push b su pila 2
   - Pop da pila 1 e push su pila 2

- Se pila 1 è vuota, push ⊔ su pila 2

3. **Stato e accettazione:**
   - Gli stati del 2-PDA corrispondono agli stati della TM
   - Accettazione quando si raggiunge uno stato finale

Correttezza: Questa costruzione preserva la configurazione del nastro e simula fedelmente ogni passo della TM.

Quindi 2-PDA ≡ TM. □

Problema 4 (9 punti)

Parte (a): PIVOT ∈ NP

Certificato: Dato input ⟨n, W⟩, il certificato è una coalizione $C \subseteq \{1, ..., n-1\}$.

Verificatore polinomiale:

```
Verifica(⟨n, W⟩, C):
1. Calcola S = Σ_{j∈C} W[j]
2. Calcola T = Σ^n_{j=1} W[j]
3. Verifica che S < T/2 < S + W[n]
4. Return true se entrambe le condizioni sono soddisfatte
```

Complessità: O(n) per calcolare le somme.

Quindi PIVOT ∈ NP. □

Parte (b): PIVOT è NP-hard

Riduzione: SET-PARTITION $\leq_p$ PIVOT

Dato: Un'istanza ⟨S⟩ di SET-PARTITION dove $S = \{s_1, s_2, ..., s_k\}$ e $T = \Sigma_i s_i$.

Costruzione:

- $n = k + 1$
- $W[i] = s_i$ per $i = 1, ..., k$
- $W[n] = W[k+1] = T/2$ (assumiamo T pari; se dispari, SET-PARTITION ha risposta NO)

Correttezza:

⇒**:** Se SET-PARTITION ha risposta YES, esistono S_1, S_2 con $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$, $\Sigma_{x \in S_1} x = \Sigma_{y \in S_2} y = T/2$.

Sia $C = \{i \mid s_i \in S_1\}$. Allora:

- $\Sigma_{j \in C} W[j] = T/2$
- $\Sigma^{n}_{j=1} W[j] = T + T/2 = 3T/2$
- $T/2 < 3T/4$ e $T/2 + T/2 = T > 3T/4$

Quindi l'elettore n è pivot.

⇐**:** Se l'elettore n è pivot, esiste C tale che:

- $\Sigma_{j \in C} W[j] < 3T/4$
- $\Sigma_{j \in C} W[j] + T/2 > 3T/4$

Dalla seconda: $\Sigma_{j \in C} W[j] > T/4$
Dalla prima: $\Sigma_{j \in C} W[j] < 3T/4$

Ma $\Sigma_{j \in C} W[j]$ è somma di elementi di S, e l'unico modo per avere una somma tra T/4 e 3T/4 che sia anche $< 3T/4$ è che sia esattamente T/2.

Quindi $\{s_i \mid i \in C\}$ è una partizione di S con somma T/2.

La riduzione è chiaramente polinomiale. Quindi PIVOT è NP-hard. □

Conclusione: PIVOT è NP-completo.