Automi e Linguaggi Formali - Esame del 15 Luglio 2024

Problema 1 (12 punti)

Parte (a): Definizione formale della funzione di transizione di una JTM

Definizione formale di JTM: Una JTM (Jumping Turing Machine) è una tupla $M = (\Gamma, \delta, w_0)$ dove:

- Γ è l'alfabeto del nastro che include sempre $\{A, E, N, Y, \sqcup\}$
- $w_0 \in \Gamma^3$ è la configurazione iniziale della testina (3 simboli)
- $\delta: \Gamma^3 \to \Gamma^3 \times \{L, R, HALT\}$ è la funzione di transizione

Funzione di transizione: $\delta(a, b, c) = (a', b', c', d)$ dove:

- $(a, b, c) \in \Gamma^3$ è il contenuto corrente delle 3 celle sotto la testina
- $(a', b', c') \in \Gamma^3$ è il nuovo contenuto da scrivere nelle 3 celle
- $d \in \{L, R, HALT\}$ è la direzione di movimento (o halt)

Condizioni speciali:

- Se $(a', b', c') = (Y, E, A)$, la macchina si ferma e accetta
- Se $(a', b', c') = (N, A, Y)$, la macchina si ferma e rifiuta
- Altrimenti, la testina si sposta di una posizione nella direzione d

Configurazione: Una configurazione è (i, tape) dove:

- i è la posizione della testina (centro delle 3 celle)
- tape: $\mathbb{Z} \to \Gamma$ è il contenuto del nastro

Parte (b): Le JTM riconoscono i linguaggi Turing-riconoscibili

Teorema: JTM ≡ TM (stessa potenza computazionale)

Dimostrazione:

Parte 1: JTM ⊆ TM Ogni JTM può essere simulata da una TM standard:

- La TM mantiene lo stesso nastro della JTM
- Simula la lettura di 3 celle consecutive usando stati per "ricordare" i simboli letti

- Simula la scrittura di 3 celle con una sequenza di scritture individuali
- Gestisce i movimenti di una cella alla volta

Parte 2: TM ⊆ JTM (costruzione principale) Data una TM M = (Q, Σ, Γ, δ, q_0, q_acc, q_rej), costruiamo una JTM equivalente.

Codifica dello stato sul nastro: Usiamo una codifica speciale dove lo stato corrente e la posizione della testina originale sono codificati direttamente sul nastro:

- Ogni cella del nastro originale è rappresentata da un blocco di simboli
- Il blocco contiene: [stato_corrente, simbolo_originale, marker_testina]

Algoritmo di simulazione:

1. **Inizializzazione:** Codifica l'input e lo stato iniziale sul nastro
2. **Ricerca della testina:** Trova il blocco marcato come posizione della testina
3. **Lettura dello stato:** Estrae stato corrente e simbolo sotto la testina
4. **Applicazione della transizione:** Applica δ(stato, simbolo) = (nuovo_stato, nuovo_simbolo, direzione)
5. **Aggiornamento:** Modifica il nastro per riflettere la nuova configurazione
6. **Movimento:** Sposta il marker della testina nella direzione appropriata
7. **Controllo terminazione:** Se raggiunge stato finale, scrive YEA o NAY

Gestione dei dettagli tecnici:

- La JTM usa la sua capacità di leggere/scrivere 3 celle per manipolare efficacemente i blocchi
- I movimenti di una cella permettono di navigare tra i blocchi
- Stati di accettazione/rifiuto della TM sono mappati in YEA/NAY

Correttezza: La simulazione preserva la semantica di ogni transizione della TM originale.

Quindi JTM ≡ TM, e le JTM riconoscono esattamente i linguaggi Turing-riconoscibili. □

Problema 2 (12 punti)

Parte (a): Formulazione come linguaggio GROSS_TM

Definizione di grawlix: Un grawlix è una stringa di simboli "volgari" (da definire). Sia GRAWLIX l'insieme di tutte le stringhe considerate grawlix.

Definizione del problema:

```
GROSS_TM = { ⟨M⟩ | M è una TM e L(M) ∩ GRAWLIX ≠ ∅ }
```

In altre parole, GROSS_TM contiene le codifiche delle TM il cui linguaggio contiene almeno un grawlix.

Parte (b): GROSS_TM è indecidibile

Teorema: GROSS_TM è indecidibile.

Dimostrazione per riduzione da A_TM:

Useremo il fatto che A_TM = {⟨M, w⟩ | M accetta w} è indecidibile.

Riduzione: A_TM ≤ GROSS_TM

Dato: Un'istanza ⟨M, w⟩ di A_TM
Costruzione: Costruiamo una TM M' tale che ⟨M'⟩ ∈ GROSS_TM ⟺ ⟨M, w⟩ ∈ A_TM

Costruzione di M':

```
M'(input x):
1. Ignora l'input x
2. Simula M su w
3. Se M accetta w:
   - Scrivi e accetta il grawlix "*#@%!"
4. Se M rifiuta w:
   - Rifiuta (non accetta nessun grawlix)
5. Se M cicla su w:
   - Cicla (non accetta nessun grawlix)
```

Correttezza della riduzione:

⇒**:** Se ⟨M, w⟩ ∈ A_TM, allora M accetta w.
Quindi M' raggiungerà il passo 3 e accetterà "*#@%!" ∈ GRAWLIX.
Pertanto L(M') ∩ GRAWLIX ≠ ∅, quindi ⟨M'⟩ ∈ GROSS_TM.

⇐**:** Se ⟨M'⟩ ∈ GROSS_TM, allora L(M') ∩ GRAWLIX ≠ ∅.
L'unico grawlix che M' può accettare è "*#@%!", e questo accade solo se M accetta w.
Quindi ⟨M, w⟩ ∈ A_TM.

Computabilità della riduzione: M' può essere costruita algoritmicamente da M e w.

Poiché A_TM è indecidibile e A_TM ≤ GROSS_TM, concludiamo che GROSS_TM è indecidibile. □

Problema 3 (12 punti)

Parte (a): ALIBABA ∈ NP

Problema ALIBABA:

$$\text{ALIBABA} = \{\langle N, P, W, M, L\rangle \mid \exists B \subseteq \{1,\ldots,N\} : \Sigma_{j\in B} W[j] \leq M \wedge \Sigma_{j\in B} P[j] \geq L\}$$

Certificato: Un sottoinsieme $B \subseteq \{1, 2, ..., N\}$

Verificatore polinomiale:

```
Verify(⟨N, P, W, M, L⟩, B):
1. peso_totale := 0
2. prezzo_totale := 0
3. Per ogni j ∈ B:
   a. peso_totale := peso_totale + W[j]
   b. prezzo_totale := prezzo_totale + P[j]
4. Return (peso_totale ≤ M) ∧ (prezzo_totale ≥ L)
```

Analisi di complessità:

- Dimensione del certificato: O(N) bit per rappresentare B
- Tempo di verifica: O(N) per calcolare le somme
- La verifica è polinomiale

Quindi ALIBABA ∈ NP. □

Parte (b): ALIBABA è NP-hard

Riduzione: SUBSET-SUM $\leq_p$ ALIBABA

Dato: Un'istanza ⟨S, t⟩ di SUBSET-SUM dove $S = \{s_1, s_2, ..., s_k\}$ e target t.

Costruzione dell'istanza ALIBABA:

- N = k (numero di oggetti = dimensione di S)

- $P[i] = s_i$ per $i = 1, ..., k$ (prezzo = valore nell'insieme)
- $W[i] = s_i$ per $i = 1, ..., k$ (peso = valore nell'insieme)
- $M = t$ (limite di peso = target)
- $L = t$ (prezzo minimo = target)

Correttezza della riduzione:

⇒: Se $\langle S, t\rangle \in$ SUBSET-SUM, allora $\exists S' \subseteq S$ tale che $\Sigma_{x \in S'} x = t$.

Sia $B = \{i \mid s_i \in S'\}$. Allora:

- $\Sigma_{j \in B} W[j] = \Sigma_{j \in B} s_j = \Sigma_{x \in S'} x = t = M$ ✓
- $\Sigma_{j \in B} P[j] = \Sigma_{j \in B} s_j = \Sigma_{x \in S'} x = t = L$ ✓

Quindi $\langle N, P, W, M, L\rangle \in$ ALIBABA.

⇐: Se $\langle N, P, W, M, L\rangle \in$ ALIBABA, allora $\exists B \subseteq \{1,...,N\}$ tale che:

- $\Sigma_{j \in B} W[j] \leq M = t$
- $\Sigma_{j \in B} P[j] \geq L = t$

Poiché $W[j] = P[j] = s_j$, abbiamo:

- $\Sigma_{j \in B} s_j \leq t$
- $\Sigma_{j \in B} s_j \geq t$

Quindi $\Sigma_{j \in B} s_j = t$, e $S' = \{s_j \mid j \in B\}$ è una soluzione per SUBSET-SUM.

Computabilità: La trasformazione è chiaramente polinomiale.

Poiché SUBSET-SUM è NP-completo e SUBSET-SUM $\leq_p$ ALIBABA, concludiamo che ALIBABA è NP-hard. □

Conclusione: ALIBABA è NP-completo.