Automi e Linguaggi Formali - 12/6/2024

Seconda prova intermedia – Secondo Turno - Soluzioni

Esercizio 1 (12 punti) - Macchina di Turing con "save e restore" (SRTM)

(a) Definizione formale della funzione di transizione di una SRTM

Una **SRTM** (Save-Restore Turing Machine) è una macchina di Turing deterministica a singolo nastro che può salvare e ripristinare configurazioni.

Definizione formale:

Una SRTM è una 7-tupla M = (Q, Σ, Γ, δ, q_0, $q_a$ccept, $q_r$eject) dove:

- Q è l'insieme finito degli stati
- Σ è l'alfabeto di input (Σ ⊂ Γ)
- Γ è l'alfabeto del nastro
- q_0 ∈ Q è lo stato iniziale
- $q_a$ccept ∈ Q è lo stato di accettazione
- $q_r$eject ∈ Q è lo stato di rifiuto
- δ: Q × Γ → Q × Γ × {L, R, SAVE, RESTORE} è la funzione di transizione estesa

Semantica operazionale:

- Una configurazione è rappresentata da (q, w, i, saved) dove:
  - q ∈ Q è lo stato corrente
  - w ∈ Γ* è il contenuto del nastro
  - i è la posizione della testina
  - saved è la configurazione salvata (o ∅ se nessuna)
- Le operazioni speciali sono:
  - **SAVE**: saved ← (q, w, i) (sovrascrive configurazione precedente)
  - **RESTORE**: se saved ≠ ∅, allora (q, w, i) ← saved; altrimenti nessun effetto

(b) Dimostrazione che le SRTM riconoscono i linguaggi Turing-riconoscibili

Teorema: Le SRTM riconoscono esattamente la classe dei linguaggi Turing-riconoscibili.

Dimostrazione:

(⊆) Ogni linguaggio riconosciuto da una SRTM è Turing-riconoscibile:

Data una SRTM M, costruiamo una TM standard N che la simula:

Descrizione implementativa di N:

```
N = "Su input w:
1. Inizializza una variabile saved = ∅
2. Simula M mantenendo traccia della configurazione corrente
3. Quando M esegue SAVE:
  - Copia la configurazione corrente in saved
4. Quando M esegue RESTORE:
  - Se saved ≠ ∅, ripristina la configurazione da saved
  - Altrimenti continua normalmente
5. Accetta se M accetta, rifiuta se M rifiuta"
```

N può rappresentare saved usando nastri multipli o codificandolo sul nastro principale.

(⊇) Ogni linguaggio Turing-riconoscibile è riconosciuto da una SRTM:

Data una TM standard M, costruiamo una SRTM N equivalente:

Descrizione implementativa di N:

```
N = "Su input w:
1. Simula esattamente M ignorando le operazioni SAVE e RESTORE
2. Accetta se M accetta, rifiuta se M rifiuta"
```

Poiché N non usa mai SAVE o RESTORE, si comporta come una TM standard. ■

Esercizio 2 (12 punti) - Linguaggio palindromo

(a) Formulazione come linguaggio PALTM

Definizione:

```
PALTM = {⟨M⟩ | M è una TM e L(M) è un linguaggio palindromo}
```

dove un linguaggio $B \subseteq \{0,1\}^*$ è palindromo se per ogni $w \in B$, $w = w^R$ (w è uguale al suo rovesciato).

(b) Dimostrazione che PALTM è indecidibile

Teorema: PALTM è indecidibile.

Dimostrazione per riduzione da ATM:

Supponiamo per contraddizione che PALTM sia decidibile e sia R un decisore per PALTM. Costruiamo un decisore S per ATM:

Descrizione implementativa di S:

```
S = "Su input ⟨M, w⟩:
1. Costruisci la seguente TM M':
   M' = "Su input x:
   a) Se x = 0: accetta
   b) Se x = 1: accetta
   c) Se x = 01: simula M su w
      - Se M accetta w: accetta
      - Se M rifiuta w: rifiuta
      - Se M va in loop: va in loop
   d) Per ogni altro input: rifiuta"

2. Esegui R su ⟨M'⟩
3. Se R accetta: rifiuta
4. Se R rifiuta: accetta"
```

Analisi della correttezza:

- Se M accetta w:
  - L(M') = {0, 1, 01}
  - Questo NON è palindromo (01 ≠ 10)
  - R rifiuta ⟨M'⟩
  - S accetta ⟨M, w⟩ ✓
- Se M non accetta w (rifiuta o va in loop):

- $L(M') = \{0, 1\}$
- Questo È palindromo ($0 = 0^R$, $1 = 1^R$)
- R accetta ⟨M'⟩
- S rifiuta ⟨M, w⟩ ✓

Quindi S decide ATM, contraddicendo l'indecidibilità di ATM. Pertanto PALTM è indecidibile. ■

Esercizio 3 (12 punti) - Problema SUPPLY

(a) Dimostrazione che SUPPLY è in NP

Teorema: SUPPLY ∈ NP.

Dimostrazione:

Costruiamo un verificatore polinomiale V per SUPPLY:

Descrizione implementativa di V:

```
V = "Su input ⟨⟨S₁, ..., Sₙ, k⟩, T⟩:
1. Verifica che T ⊆ {1, 2, ..., n}
2. Verifica che |T| = k
3. Calcola U = ∪ᵢ∈T Sᵢ
4. Verifica che U = {1, 2, ..., m}
5. Se tutti i controlli passano: accetta
6. Altrimenti: rifiuta"
```

- Il certificato T ha dimensione $O(k) \leq O(n)$
- Tutti i controlli richiedono tempo polinomiale
- V accetta $\langle\langle S_1, \ldots, S_n, k\rangle, T\rangle \iff$ T è una fornitura valida di dimensione k

Quindi SUPPLY ∈ NP. ■

(b) Dimostrazione che SUPPLY è NP-hard

Teorema: SUPPLY è NP-hard.

Dimostrazione per riduzione da VERTEX-COVER:

Data un'istanza $\langle G, k\rangle$ di VERTEX-COVER dove $G = (V, E)$, costruiamo un'istanza di SUPPLY:

Costruzione della riduzione f:

```
f(⟨G, k⟩) = ⟨S₁, S₂, ..., S|V|, k⟩
```

dove:

- Ogni vertice $v_i \in V$ corrisponde a un fornitore i
- Ogni arco $e_j \in E$ corrisponde a un ingrediente j
- $S_i = \{j \mid e_j \text{ è incidente al vertice } v_i\}$

Dimostrazione di correttezza:

(⟹) Se G ha una copertura di vertici T di dimensione k:

- Ogni arco e_j ha almeno un estremo in T
- Quindi $\bigcup_{i \in T} S_i$ contiene tutti gli ingredienti $\{1, 2, ..., |E|\}$
- T è una fornitura valida di dimensione k per l'istanza di SUPPLY

(⟸) Se esiste una fornitura valida T di dimensione k:

- $\bigcup_{i \in T} S_i = \{1, 2, ..., |E|\}$
- Per ogni arco e_j, esiste almeno un $v_i \in T$ tale che $j \in S_i$
- Questo significa che e_j è incidente a v_i
- Quindi T è una copertura di vertici di dimensione k per G

Complessità: La riduzione opera in tempo $O(|V| + |E|)$, che è polinomiale.

Quindi VERTEX-COVER $\leq_p$ SUPPLY, e poiché VERTEX-COVER è NP-completo, SUPPLY è NP-hard. ■

Conclusione: SUPPLY è NP-completo (in NP e NP-hard).