Soluzione Appello Automi e Linguaggi Formali - 10/6/2025

1. **(12 punti)** Una *Macchina di Turing con salto a destra (RJTM)* è una variante della macchina di Turing che estende il modello standard introducendo un meccanismo per accedere direttamente a una qualsiasi posizione del nastro, senza dover scorrere sequenzialmente le celle. Una RJTM è dotata di due nastri:

   - un nastro di lavoro dove può leggere, scrivere e spostarsi a piacere. All'inizio della computazione, questo nastro contiene l'input;
   - un nastro puntatore con alfabeto binario. Anche in questo nastro la macchina può leggere, scrivere e spostarsi a piacere.

   Oltre alle consuete operazioni di lettura, scrittura e spostamento delle testine, una RJTM può eseguire un'operazione aggiuntiva di salto a destra. Per eseguire questa operazione, la macchina legge il numero binario p sul nastro puntatore e poi sposta la testina sul nastro di lavoro a destra di p celle.

   (a) Dai una definizione formale della funzione di transizione di una RJTM.

   (b) Dimostra che le RJTM riconoscono la classe dei linguaggi Turing-riconoscibili. Usa una descrizione a livello implementativo per definire le macchine di Turing.

2. **(12 punti)** Data una parola w su un alfabeto Σ, si dice che $u \in \Sigma^*$ è un prefisso di w se esiste una stringa $v \in \Sigma^*$ tale che $w = uv$. Un linguaggio $L \subseteq \Sigma^*$ è *chiuso per prefisso* se per ogni parola $w \in L$, tutti i prefissi u di w appartengono anch'essi a L. Considera il problema di determinare se il linguaggio di una TM M è chiuso per prefisso.

   (a) Formula questo problema come un linguaggio $\textsc{PrefixClosed}_{TM}$.

   (b) Dimostra che il linguaggio $\textsc{PrefixClosed}_{TM}$ è indecidibile.

3. **(12 punti)** Un sottoinsieme S di vertici di un grafo $G = (V, E)$ è una *quasi copertura* se esiste esattamente un arco di G che non ha estremi in S. Considera il seguente problema:

   $$\textsc{AlmostVertexCover} = \{\langle G, k\rangle \mid \text{esiste } S \subseteq V \text{ quasi copertura di cardinalità } k\}$$

   (a) Dimostra che $\textsc{AlmostVertexCover}$ è un problema NP.

   (b) Dimostra che $\textsc{AlmostVertexCover}$ è NP-hard, usando $\textsc{VertexCover}$ come problema NP-hard di riferimento.

Problema 1: RJTM (Right Jump Turing Machine) - 12 punti

(a) Definizione formale della funzione di transizione

Una RJTM è caratterizzata da:

- **Q**: insieme finito di stati
- **Σ**: alfabeto di input
- **Γ**: alfabeto del nastro di lavoro (Σ ⊆ Γ)
- **{0,1,␣}**: alfabeto del nastro puntatore

- ⊔: simbolo blank
- $q_0 \in Q$: stato iniziale
- $q_{accept}, q_{rifiuto} \in Q$: stati di accettazione e rifiuto

La **funzione di transizione** è definita come:

$$\delta: Q \times \Gamma \times \{0,1,\sqcup\} \rightarrow Q \times \Gamma \times \{L,R,S\} \times \{0,1,\sqcup\} \times \{L,R,S\} \times \{J\}$$

dove:

- Il primo Γ è il simbolo letto dal nastro di lavoro
- Il primo {0,1,⊔} è il simbolo letto dal nastro puntatore
- Il secondo Γ è il simbolo scritto sul nastro di lavoro
- Il primo {L,R,S} è il movimento della testina del nastro di lavoro
- Il secondo {0,1,⊔} è il simbolo scritto sul nastro puntatore
- Il secondo {L,R,S} è il movimento della testina del nastro puntatore
- {J} indica se eseguire l'operazione di salto a destra (opzionale)

Operazione di salto a destra: Se δ include J, la macchina:

1. Legge il numero binario p dal nastro puntatore (dall'inizio fino al primo ⊔)
2. Sposta la testina del nastro di lavoro a destra di p posizioni

(b) Dimostrazione dell'equivalenza con i linguaggi Turing-riconoscibili

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

Dimostrazione:

Direzione 1: Ogni linguaggio Turing-riconoscibile è riconosciuto da una RJTM.

Questa direzione è banale. Data una TM standard M che riconosce un linguaggio L, costruiamo una RJTM R equivalente che:

- Usa solo il nastro di lavoro (ignorando il nastro puntatore)
- Non esegue mai operazioni di salto
- Simula esattamente M sul nastro di lavoro

Direzione 2: Ogni linguaggio riconosciuto da una RJTM è Turing-riconoscibile.

Data una RJTM R, costruiamo una TM standard M a 3 nastri che simula R:

M = "Su input w:

1. **Inizializzazione**: Copia w sul primo nastro (nastro di lavoro), lascia vuoti il secondo nastro (nastro puntatore) e usa il terzo nastro per operazioni ausiliarie.

2. **Simulazione delle operazioni standard**: Per $\delta(q,a,b) = (r,c,d_1,e,d_2)$ senza salto:
   - Scrivi c sulla cella corrente del primo nastro
   - Scrivi e sulla cella corrente del secondo nastro
   - Muovi le testine secondo d_1 e d_2
   - Passa allo stato r
3. **Simulazione del salto a destra**: Per $\delta(q,a,b) = (r,c,d_1,e,d_2,J)$:
   - Esegui le operazioni standard del punto 2
   - **Calcolo del salto**:
     - Sposta la testina del secondo nastro all'inizio
     - Leggi la sequenza binaria fino al primo ⊔ e calcola p
     - Sposta la testina del primo nastro di p posizioni a destra
   - Continua la simulazione nello stato r
4. **Terminazione**: Se R raggiunge q_{accept}, M accetta. Se R raggiunge $q_{rifiuto}$, M rifiuta.

La simulazione è corretta perché ogni operazione di R può essere implementata in tempo finito da M usando i suoi tre nastri. □

Problema 2: Linguaggi chiusi per prefisso - 12 punti

(a) Formulazione del problema PrefixClosedTM

PrefixClosedTM = {⟨M⟩ | M è una TM e L(M) è chiuso per prefisso}

dove un linguaggio $L \subseteq \Sigma^*$ è chiuso per prefisso se: $\forall w \in L$, $\forall u$ prefisso di $w \Rightarrow u \in L$

(b) Dimostrazione dell'indecidibilità

Teorema: PrefixClosedTM è indecidibile.

Dimostrazione: Usiamo una riduzione da A_TM ≤_m PrefixClosedTM.

Costruiamo la seguente funzione calcolabile f:

f = "Su input ⟨M,w⟩:

1. Costruisci la seguente TM M':

   M' = "Su input x:
   1. Se x = ε, accetta
   2. Se x ha la forma 0^i per qualche i ≥ 1:
      - Simula M su w per i passi
      - Se M accetta w entro i passi, accetta x
      - Altrimenti, rifiuta x

    3. Per ogni altro input, rifiuta
2. Restituisci ⟨M'⟩"

Analisi della riduzione:

Caso 1: Se ⟨M,w⟩ ∈ A_TM (M accetta w):

- M accetta w in un numero finito k di passi
- L(M') = {ε, 0, 0², ..., 0^k}
- Questo linguaggio è chiuso per prefisso
- Quindi ⟨M'⟩ ∈ PrefixClosedTM

Caso 2: Se ⟨M,w⟩ ∉ A_TM (M non accetta w):

- M non accetta mai w (loop infinito o rifiuto)
- L(M') = {ε}
- Il linguaggio {ε} è chiuso per prefisso
- Quindi ⟨M'⟩ ∈ PrefixClosedTM

Problema: La riduzione non funziona perché in entrambi i casi il linguaggio risulta chiuso per prefisso.

Correzione della riduzione:

M' = "Su input x:

1. Se x = ε, accetta
2. Se x = 0^i10^j per i,j ≥ 0:
    - Simula M su w per i passi
    - Se M accetta w entro i passi, accetta x
    - Altrimenti, rifiuta x
3. Per ogni altro input, rifiuta"

Analisi corretta:

Caso 1: Se M accetta w in k passi:

- L(M') = {ε} ∪ {0^i10^j | i ≥ k, j ≥ 0}
- La stringa 0^k1 ∈ L(M') ma il suo prefisso 0^(k-1) ∉ L(M')
- Quindi L(M') non è chiuso per prefisso
- ⟨M'⟩ ∉ PrefixClosedTM

Caso 2: Se M non accetta w:

- L(M') = {ε}
- Questo linguaggio è chiuso per prefisso

- ⟨M'⟩ ∈ PrefixClosedTM

Quindi f è una riduzione valida e PrefixClosedTM è indecidibile. □

Problema 3: AlmostVertexCover - 12 punti

(a) Dimostrazione che AlmostVertexCover ∈ NP

Teorema: AlmostVertexCover ∈ NP.

Dimostrazione: Definiamo un algoritmo di verifica polinomiale.

Certificato: Un insieme S ⊆ V di cardinalità k.

Algoritmo di verifica V:

```
V = "Su input ⟨⟨G,k⟩, S⟩:
1. Verifica che |S| = k
2. Conta gli archi non coperti da S:
   uncovered = 0
   Per ogni arco (u,v) ∈ E:
     Se u ∉ S e v ∉ S:
       uncovered++
3. Se uncovered = 1, accetta
4. Altrimenti, rifiuta"
```

Correttezza:

- Se ⟨G,k⟩ ∈ AlmostVertexCover, esiste S con |S| = k che è quasi copertura, quindi V accetta con certificato S
- Se V accetta con certificato S, allora S è una quasi copertura di cardinalità k, quindi ⟨G,k⟩ ∈ AlmostVertexCover

Complessità: V esegue O(|E|) operazioni, quindi tempo polinomiale.

Pertanto AlmostVertexCover ∈ NP. □

(b) Dimostrazione che AlmostVertexCover è NP-hard

Teorema: AlmostVertexCover è NP-hard.

Dimostrazione: Riduciamo VertexCover ≤_p AlmostVertexCover.

Data un'istanza ⟨G=(V,E), k⟩ di VertexCover, costruiamo:

f(⟨G,k⟩) = ⟨G'=(V',E'), k+1⟩

dove:

- V' = V ∪ {u_0, v_0} (aggiungiamo due nuovi vertici)
- E' = E ∪ {(u_0,v_0)} (aggiungiamo un nuovo arco)

Analisi della riduzione:

Caso 1: Se ⟨G,k⟩ ∈ VertexCover:

- Esiste S ⊆ V con |S| = k che copre tutti gli archi di E
- Consideriamo S' = S ∪ {u_0} ⊆ V'
- |S'| = k + 1
- S' copre tutti gli archi di E (perché S li copriva)
- S' non copre l'arco (u_0,v_0) perché v_0 ∉ S'
- Quindi S' è una quasi copertura di G' con cardinalità k+1
- ⟨G',k+1⟩ ∈ AlmostVertexCover

Caso 2: Se ⟨G,k⟩ ∉ VertexCover:

- Ogni S ⊆ V con |S| = k lascia scoperti almeno 2 archi di E
- Consideriamo qualsiasi S' ⊆ V' con |S'| = k+1

Sottocaso 2a: Se {u_0,v_0} ⊆ S':

- S' contiene al più k-1 vertici di V
- Quindi S' ∩ V lascia scoperti almeno 2 archi di E
- S' non è una quasi copertura

Sottocaso 2b: Se |S' ∩ {u_0,v_0}| ≤ 1:

- S' contiene al più k vertici di V
- S' ∩ V lascia scoperti almeno 2 archi di E
- Inoltre, se nessuno tra u_0,v_0 è in S', anche (u_0,v_0) è scoperto
- S' non è una quasi copertura

Sottocaso 2c: Se esattamente uno tra u_0,v_0 è in S':

- S' contiene al più k vertici di V
- S' ∩ V lascia scoperti almeno 2 archi di E
- L'arco (u_0,v_0) è scoperto
- Totale: almeno 3 archi scoperti
- S' non è una quasi copertura

In tutti i sottocasi, nessun insieme di cardinalità k+1 è una quasi copertura. Quindi ⟨G',k+1⟩ ∉ AlmostVertexCover.

La riduzione è polinomiale (aggiunge solo 2 vertici e 1 arco) e corretta. Quindi AlmostVertexCover è NP-hard. □

Conclusione: AlmostVertexCover è NP-completo. □

Esercizio 1 - RATM

1. **(12 punti)** Una *Macchina di Turing ad accesso casuale (RATM)* è una variante della macchina di Turing che estende il modello standard introducendo un meccanismo per accedere direttamente a una qualsiasi posizione dell'input, senza dover scorrere sequenzialmente le celle del nastro. Una RATM è dotata di tre nastri:
   - un nastro di input a sola lettura, che contiene l'input;
   - un nastro di lavoro dove la macchina può leggere, scrivere e spostarsi a piacere;
   - un nastro puntatore con alfabeto binario. Anche in questo nastro la macchina può leggere, scrivere e spostarsi a piacere.

   Oltre alle consuete operazioni di lettura, scrittura e spostamento delle testine, una RATM può eseguire un'operazione aggiuntiva di accesso diretto all'input. Per eseguire questa operazione, la macchina legge il numero binario p sul nastro puntatore e poi scrive il p-esimo simbolo dell'input sulla cella corrente del nastro lavoro. I simboli dell'input sono numerati da sinistra a destra a partire dalla posizione 0.

   (a) Dai una definizione formale della funzione di transizione di una RATM.

   (b) Dimostra che le RATM riconoscono la classe dei linguaggi Turing-riconoscibili. Usa una descrizione a livello implementativo per definire le macchine di Turing.

Dimostrazione nastro singolo

(a) Definizione formale della funzione di transizione

$\delta : Q \times \Sigma \times \Gamma \times \{0,1,\sqcup\} \to Q \times \Gamma \times \{L,R,S\} \times \{0,1,\sqcup\} \times \{L,R,S\} \times \{L,R,S\} \times \{A\}$

dove:

- Il primo Σ è il simbolo letto dal nastro di input
- Il primo Γ è il simbolo letto dal nastro di lavoro
- Il primo {0,1,⊔} è il simbolo letto dal nastro puntatore
- Il secondo Γ è il simbolo scritto sul nastro di lavoro
- Il primo {L,R,S} è il movimento della testina del nastro di lavoro
- Il secondo {0,1,⊔} è il simbolo scritto sul nastro puntatore
- Il secondo {L,R,S} è il movimento della testina del nastro puntatore
- Il terzo {L,R,S} è il movimento della testina del nastro di input
- A indica l'operazione di accesso diretto all'input (opzionale)

Operazione di accesso diretto: Se δ include A, la macchina:

1. Legge il numero binario p dal nastro puntatore (dall'inizio fino al primo ⊔)
2. Scrive il p-esimo simbolo dell'input sulla cella corrente del nastro di lavoro

(b) Dimostrazione dell'equivalenza con i linguaggi Turing-riconoscibili

Per dimostrare che le RATM riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose: che ogni linguaggio Turing-riconoscibile è riconosciuto da una RATM, e che ogni linguaggio riconosciuto da una RATM è Turing-riconoscibile.

Prima dimostrazione: La prima dimostrazione è banale: le TM deterministiche a singolo nastro sono un caso particolare di RATM che utilizzano solo il nastro di lavoro, non accedono mai al nastro di input dopo la fase iniziale e non effettuano mai l'operazione di accesso diretto. Di conseguenza, ogni linguaggio Turing-riconoscibile è riconosciuto da una RATM.

Seconda dimostrazione: Per dimostrare che ogni linguaggio riconosciuto da una RATM è Turing-riconoscibile, mostriamo come convertire una RATM M in una TM deterministica a nastro singolo S equivalente.

S = "Su input w:

1. Inizializza il nastro con una rappresentazione che contiene: l'input w, uno spazio di lavoro, uno spazio per il nastro puntatore e marcatori per separare le sezioni. La configurazione iniziale è `#w` ##⊔#, dove i simboli # separano le diverse sezioni.
2. Per simulare le operazioni standard di lettura, scrittura e movimento sui nastri di lavoro e puntatore, S scorre il nastro per raggiungere la sezione appropriata, esegue l'operazione richiesta e aggiorna le posizioni delle testine usando marcature speciali.
3. Per simulare una mossa del tipo $\delta(q,a,b,c) = (r,d,m_1,e,m_2,m_3)$ senza accesso diretto:
   - S identifica la posizione corrente su ciascun nastro simulato
   - Scrive d nella sezione di lavoro secondo m_1
   - Scrive e nella sezione puntatore secondo m_2
   - Muove la marcatura del nastro di input secondo m_3
   - Passa allo stato r
4. Per simulare una mossa del tipo $\delta(q,a,b,c) = (r,d,m_1,e,m_2,m_3,A)$ con accesso diretto:
   - Esegui le operazioni del punto 3
   - Sposta la testina all'inizio della sezione puntatore
   - Leggi la sequenza binaria fino al primo ⊔ e calcola il valore decimale p
   - Accedi alla p-esima posizione dell'input (se esiste) e copia il simbolo nella posizione corrente della sezione di lavoro
   - Continua la simulazione nello stato r
5. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora S termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di

rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

La simulazione è corretta perché ogni operazione della RATM può essere implementata in tempo finito dalla TM a nastro singolo, incluso il calcolo del valore binario e l'accesso diretto all'input.

Dimostrazione multinastro

(a) Definizione formale della funzione di transizione

$\delta : Q \times \Sigma \times \Gamma \times \{0,1,\sqcup\} \to Q \times \Gamma \times \{L,R,S\} \times \{0,1,\sqcup\} \times \{L,R,S\} \times \{L,R,S\} \times \{A\}$

dove:

- Il primo Σ è il simbolo letto dal nastro di input
- Il primo Γ è il simbolo letto dal nastro di lavoro
- Il primo $\{0,1,\sqcup\}$ è il simbolo letto dal nastro puntatore
- Il secondo Γ è il simbolo scritto sul nastro di lavoro
- Il primo $\{L,R,S\}$ è il movimento della testina del nastro di lavoro
- Il secondo $\{0,1,\sqcup\}$ è il simbolo scritto sul nastro puntatore
- Il secondo $\{L,R,S\}$ è il movimento della testina del nastro puntatore
- Il terzo $\{L,R,S\}$ è il movimento della testina del nastro di input
- A indica l'operazione di accesso diretto all'input (opzionale)

Operazione di accesso diretto: Se δ include A, la macchina:

1. Legge il numero binario p dal nastro puntatore (dall'inizio fino al primo $\sqcup$)
2. Scrive il p-esimo simbolo dell'input sulla cella corrente del nastro di lavoro

(b) Dimostrazione dell'equivalenza con i linguaggi Turing-riconoscibili

Per dimostrare che le RATM riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose: che ogni linguaggio Turing-riconoscibile è riconosciuto da una RATM, e che ogni linguaggio riconosciuto da una RATM è Turing-riconoscibile.

Prima dimostrazione: La prima dimostrazione è banale: le TM deterministiche a singolo nastro sono un caso particolare di RATM che utilizzano solo il nastro di lavoro, non accedono mai al nastro di input dopo la fase iniziale e non effettuano mai l'operazione di accesso diretto. Di conseguenza, ogni linguaggio Turing-riconoscibile è riconosciuto da una RATM.

Seconda dimostrazione: Per dimostrare che ogni linguaggio riconosciuto da una RATM è Turing-riconoscibile, mostriamo come convertire una RATM M in una TM deterministica a tre

nastri S equivalente.

S = "Su input w:

1. Inizializza i tre nastri: il primo nastro contiene l'input w (simulazione del nastro di input), il secondo nastro è vuoto (simulazione del nastro di lavoro), e il terzo nastro è vuoto (simulazione del nastro puntatore).
2. Per simulare una mossa del tipo $\delta(q,a,b,c) = (r,d,m_1,e,m_2,m_3)$ senza accesso diretto:
   - Leggi il simbolo a dal primo nastro, b dal secondo nastro, c dal terzo nastro
   - Scrivi d sulla cella corrente del secondo nastro
   - Scrivi e sulla cella corrente del terzo nastro
   - Muovi la testina del secondo nastro secondo m_1
   - Muovi la testina del terzo nastro secondo m_2
   - Muovi la testina del primo nastro secondo m_3
   - Passa allo stato r
3. Per simulare una mossa del tipo $\delta(q,a,b,c) = (r,d,m_1,e,m_2,m_3,A)$ con accesso diretto:
   - Esegui le operazioni standard del punto 2
   - **Operazione di accesso diretto**:
     - Salva la posizione corrente della testina del terzo nastro
     - Sposta la testina del terzo nastro all'inizio
     - Leggi la sequenza binaria dal terzo nastro fino al primo ⊔ e calcola il valore decimale p
     - Salva la posizione corrente della testina del primo nastro
     - Sposta la testina del primo nastro alla posizione p (se p è valido)
     - Leggi il simbolo in posizione p e scrivilo sulla cella corrente del secondo nastro
     - Ripristina le posizioni salvate delle testine del primo e terzo nastro
   - Continua la simulazione nello stato r
4. Se in qualsiasi momento la simulazione raggiunge lo stato di accettazione di M, allora S termina con accettazione. Se in qualsiasi momento la simulazione raggiunge lo stato di rifiuto di M, allora S termina con rifiuto. Negli altri casi continua la simulazione dal punto 2."

La simulazione è corretta perché ogni operazione della RATM può essere implementata in tempo finito dalla TM a tre nastri, incluso il calcolo del valore binario e l'accesso diretto all'input mediante salvataggio e ripristino delle posizioni delle testine.