Tutorato di Automi e Linguaggi Formali

Homework 8: Varianti delle Macchine di Turing e Decidibilità

Gabriel Rovesti
Corso di Laurea in Informatica - Università degli Studi di Padova

Tutorato 8 - 12-05-2025

1 Varianti Avanzate delle Macchine di Turing

Esercizio 1. Una macchina di Turing a nastri multipli è una TM con k nastri, ciascuno con la propria testina. Formalmente, viene definita come una 7-tupla $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, dove tutti i componenti sono definiti come per una TM standard, eccetto la funzione di transizione:

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

a) Dimostrate che una TM a k nastri è equivalente a una TM standard a singolo nastro in termini di potere computazionale, fornendo una costruzione dettagliata della simulazione.

b) Analizzate la complessità temporale della simulazione. Se una TM a k nastri opera in tempo $T(n)$, qual è il tempo richiesto dalla TM a nastro singolo che la simula?

c) Fornite un esempio concreto di un problema che può essere risolto più efficientemente usando una TM a nastri multipli rispetto a una TM a nastro singolo.

Esercizio 2. Una macchina di Turing a nastro circolare è una TM in cui il nastro forma un anello, permettendo alla testina di muoversi continuamente verso destra (o sinistra) senza mai raggiungere la fine del nastro. La lunghezza del nastro è finita ma può essere espansa quando necessario.

a) Dimostrate che una TM a nastro circolare è equivalente a una TM standard.

b) Descrivete un algoritmo che permetta a una TM a nastro circolare di simulare una TM standard.

c) Ideate un algoritmo per il riconoscimento di stringhe palindrome che sfrutti l'architettura a nastro circolare in modo efficiente.

Esercizio 3. Una macchina di Turing con "ferma" invece di "sinistra" è simile a una macchina di Turing ordinaria, ma la funzione di transizione ha la forma $\delta : Q \times \Gamma \to Q \times \Gamma \times \{S, R\}$. In ogni punto, la macchina può spostare la testina a destra (R) o lasciarla nella stessa posizione (S).

a) Dimostrate che questa variante della macchina di Turing è meno potente della TM standard, ovvero esiste un linguaggio riconoscibile da una TM standard che non può essere riconosciuto da una TM con "ferma" invece di "sinistra".

b) Caratterizzate la classe di linguaggi riconoscibili da una TM con "ferma" invece di "sinistra" in termini di una classe nota di linguaggi.

c) Fornite un esempio di un linguaggio semplice che può essere riconosciuto da una TM standard ma non da questa variante.

Esercizio 4. Una macchina di Turing a singola scrittura è una TM a nastro singolo che può modificare ogni cella del nastro al più una volta (inclusa la parte di input del nastro).

a) Dimostrate che questa variante di macchina di Turing è equivalente alla macchina di Turing standard, fornendo una costruzione dettagliata.

b) Descrivete un algoritmo efficiente per simulare una TM standard usando una TM a singola scrittura.

c) Discutete quali limitazioni pratiche questa restrizione impone all'implementazione di algoritmi ed eventualmente come possono essere superate.

2 Decidibilità di Problemi su Automi e Grammatiche

Esercizio 5. Sia $INFINITE_{PDA} = \{\langle P \rangle \mid P \text{ è un PDA ed } L(P) \text{ è un linguaggio infinito}\}$.

a) Dimostrate che $INFINITE_{PDA}$ è decidibile, fornendo un algoritmo dettagliato.

b) Descrivete come l'algoritmo utilizza la proprietà di pumping per i linguaggi context-free per decidere se un linguaggio è infinito.

c) Confrontate questo problema con il caso dei linguaggi regolari. Come si può decidere se un DFA riconosce un linguaggio infinito?

Esercizio 6. Dato un automa a pila (PDA), definiamo uno stato inutile come uno stato che non viene mai inserito in alcuna computazione che accetta una stringa di input. Sia $USELESS_{PDA} = \{\langle P \rangle \mid P \text{ è un PDA che ha almeno uno stato inutile}\}$.

a) Formalizzate la definizione di stato inutile in un PDA.

b) Dimostrate che $USELESS_{PDA}$ è decidibile, fornendo un algoritmo per determinare quali stati di un PDA sono inutili.

c) Discutete come l'algoritmo proposto potrebbe essere utilizzato per ottimizzare un PDA rimuovendo tutti gli stati inutili.

Esercizio 7. Sia $REV_{DFA} = \{\langle A \rangle \mid A$ è un DFA che accetta w^R in qualsiasi caso accetti $w\}$, dove w^R indica il reverse di w.

a) Dimostrate che REV_{DFA} è decidibile, fornendo un algoritmo dettagliato.

b) Caratterizzate la classe di linguaggi riconosciuti da DFA per cui vale questa proprietà.

c) Fornite un esempio non banale di un linguaggio regolare che appartiene a questa classe e uno che non vi appartiene.

3 Decidibilità di Algoritmi e Problemi Computazionali

Esercizio 8. Dato un algoritmo ricorsivo in pseudocodice, consideriamo il problema di determinare se l'algoritmo termina per ogni possibile input. Sia $TERM_{RC} = \{\langle A \rangle \mid A$ è un algoritmo ricorsivo che termina per ogni input$\}$.

a) Descrivete un insieme di condizioni sufficienti a garantire che un algoritmo ricorsivo termini per ogni input.

b) Date queste condizioni, progettate un algoritmo che analizza un algoritmo ricorsivo per verificare se soddisfa tali condizioni.

c) Discutete i limiti dell'approccio proposto, specificando tipi di algoritmi ricorsivi per cui il vostro metodo potrebbe non funzionare.

Esercizio 9. Consideriamo il problema di determinare se un grafo diretto aciclico (DAG) è un albero. Sia $TREE_{DAG} = \{\langle G \rangle \mid G$ è un DAG che è anche un albero$\}$.

a) Formalizzate la definizione di albero nel contesto dei grafi diretti.

b) Dimostrate che $TREE_{DAG}$ è decidibile, fornendo un algoritmo dettagliato.

c) Analizzate la complessità temporale e spaziale dell'algoritmo proposto.

Esercizio 10. Dato un sistema di equazioni lineari $Ax = b$, consideriamo il problema di determinare se il sistema ha almeno una soluzione. Sia $SOLVABLE_{LE} = \{\langle A, b \rangle \mid A$ è una matrice e b è un vettore tali che il sistema $Ax = b$ ha almeno una soluzione$\}$.

a) Dimostrate che $SOLVABLE_{LE}$ è decidibile, descrivendo un algoritmo basato sull'eliminazione gaussiana.

b) Analizzate la complessità temporale dell'algoritmo proposto.

c) Estendete l'algoritmo per decidere se il sistema ha esattamente una soluzione, infinite soluzioni o nessuna soluzione.