Esercizio 1

Considera il linguaggio $L = \{1^n 0^{2^n} \mid n \leq 0\}$.

Dimostra che L non è regolare.

Soluzione

Per dimostrare che il linguaggio L non è regolare, possiamo utilizzare il pumping lemma per i linguaggi regolari. Il pumping lemma afferma che se un linguaggio L è regolare, allora esiste un numero intero positivo p tale che qualsiasi stringa w in L con lunghezza maggiore o uguale a p può essere divisa in tre parti: w = xyz, dove |xy| ≤ p, |y| ≥ 1, e per ogni intero non negativo i, la stringa xy^iz appartiene a L.

Supponiamo per assurdo che il linguaggio L sia regolare. Sia p il numero intero positivo che soddisfa il pumping lemma per il linguaggio L. Consideriamo la stringa w = 1^p 0^(2^p) ┤, che appartiene a L e ha lunghezza maggiore o uguale a p.

Per il pumping lemma, la stringa w può essere scritta come w = xyz, dove |xy| ≤ p e |y| ≥ 1. Poiché la parte "1^n" della stringa w ha lunghezza p, la parte "0^(2^n)" deve essere contenuta interamente nella stringa y. Quindi, y contiene almeno una sequenza di "0" di lunghezza 2^k, dove k è un intero positivo.

Consideriamo ora la stringa xy^2z. Poiché y contiene almeno una sequenza di "0" di lunghezza 2^k, allora la stringa xy^2z contiene una sequenza di "0" di lunghezza 2^(k+1), che non appartiene a L. Infatti, l'esponente del 2 nella parte "0^(2^n)" della stringa w è sempre maggiore o uguale a n, mentre l'esponente del 2 nella parte "0^(2^(k+1))" di xy^2z è uguale a k+1, che è maggiore di n per ogni n.

Pertanto, la stringa xy^2z non appartiene a L, il che contraddice il pumping lemma. Concludiamo quindi che il linguaggio L non può essere regolare.

Esercizio 2

Chiamiamo k-PDA un automa a pila dotato di k pile. In particolare, uno 0-PDA è un NFA e un 1-PDA è un PDA convenzionale. Sappiamo già che gli 1-PDA sono più potenti degli 0-PDA (nel senso che riconoscono una classe più ampia di linguaggi). Mostra che i 3-PDA sono più potenti degli 1-PDA.

Soluzione

Per dimostrare che i 3-PDA sono più potenti degli 1-PDA, mostreremo che esiste un linguaggio che può essere riconosciuto da un 3-PDA ma non da un 1-PDA.

Consideriamo il linguaggio L = {a^n b^n c^n | n ≥ 0}. Questo linguaggio non è regolare e può essere dimostrato utilizzando il pumping lemma per i linguaggi regolari.

Mostreremo che il linguaggio L può essere riconosciuto da un 3-PDA ma non da un 1-PDA. Per farlo, costruiremo un 3-PDA che riconosce L e mostreremo che non esiste un 1-PDA equivalente.

Costruiamo un 3-PDA M che riconosce L come segue:

1. Inizialmente, M inizia in uno stato q0 e spinge un simbolo speciale # su ogni pila.

2. Quando M legge un'occorrenza del simbolo a, spinge un simbolo A su ogni pila.

3. Quando M legge un'occorrenza del simbolo b, spinge un simbolo B su ogni pila.

4. Quando M legge un'occorrenza del simbolo c, per ogni pila, effettua la seguente sequenza di operazioni:

   - Toglie un simbolo A, un simbolo B e il simbolo # dalla cima della pila.

- Se la pila è vuota, accetta. Altrimenti, torna allo stato q0.

Notare che questo 3-PDA utilizza tre pile, ognuna delle quali memorizza il conteggio di un tipo diverso di simbolo. Inoltre, per ogni pila, il 3-PDA controlla che i simboli A, B e # vengano tolti in modo coerente, in modo da assicurarsi che il numero di simboli a, b e c sia uguale.

Mostriamo ora che non esiste un 1-PDA equivalente a M. Supponiamo per assurdo che esiste un 1-PDA che riconosce L e che accetta il linguaggio L con un numero finito di stati e pile. Consideriamo la stringa w = a^p b^p c^p, dove p è un intero positivo.

Poiché l'1-PDA in questione ha solo un numero finito di stati e pile, e la stringa w ha una lunghezza maggiore di questo numero, per il principio della cassaforte, deve esistere un sottostringa x di w tale che x è accettata dal 1-PDA e può essere pompata. Tuttavia, questo implica che la sottostringa x non può contenere sia simboli a che simboli c, il che contraddice il fatto che L contiene solo stringhe della forma a^n b^n c^n.

Concludiamo quindi che non esiste un 1-PDA che riconosce il linguaggio L. Pertanto, i 3-PDA sono più potenti degli 1-PDA.

Esercizio 3

3. (9 punti) Una *macchina di Turing ad albero binario* usa un albero binario infinito come nastro, dove ogni cella nel nastro ha un figlio sinistro e un figlio destro. Ad ogni transizione, la testina si sposta dalla cella corrente al padre, al figlio sinistro oppure al figlio destro della cella corrente. Pertanto, la funzione di transizione di una tale macchina ha la forma

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{P, L, R\},$$

dove P indica lo spostamento verso il padre, L verso il figlio sinistro e R verso il figlio destro. La stringa di input viene fornita lungo il ramo sinistro dell'albero.

Mostra che qualsiasi macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

Soluzione

Dimostriamo che qualsiasi macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

Sia M una macchina di Turing ad albero binario. Possiamo simulare M utilizzando una macchina di Turing standard che utilizza un nastro lineare.

Per simulare l'albero binario di M, utilizziamo un'enumerazione depth-first delle celle dell'albero. In particolare, ogni cella dell'albero binario viene visitata in modo depth-first in un ordine predefinito, in modo che ogni cella venga visitata prima del suo figlio sinistro e del suo figlio destro. In questo modo, possiamo rappresentare l'albero binario di M come una sequenza di celle visitate in ordine depth-first.

Utilizziamo quindi un nastro lineare per simulare l'albero binario di M. In particolare, utilizziamo una sequenza di celle consecutive sul nastro per rappresentare l'albero binario, dove ogni cella contiene il simbolo che rappresenta il contenuto della cella corrispondente nell'albero binario di M, insieme a un indicatore che indica se la cella corrente è il figlio sinistro o destro della sua cella padre.

Per simulare le transizioni di M, utilizziamo una funzione di transizione che mappa ogni stato di M e ogni simbolo sul nastro lineare in uno stato e un simbolo successivi, insieme a un'operazione di movimento della testina sulla nastro lineare.

In particolare, per ogni transizione di M, esaminiamo la cella corrente sul nastro lineare e il simbolo che essa contiene, e quindi applichiamo la funzione di transizione di M per ottenere il nuovo stato e il nuovo

simbolo da scrivere sulla cella corrente, insieme all'operazione di movimento della testina sulla sequenza di celle sul nastro lineare.

In questo modo, possiamo simulare l'esecuzione di M utilizzando una macchina di Turing standard con un nastro lineare. Poiché ogni operazione della macchina di Turing ad albero binario corrisponde a un'operazione equivalente sulla macchina di Turing standard, segue che qualsiasi macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

Concludiamo che le macchine di Turing ad albero binario non aumentano il potere computazionale rispetto alle macchine di Turing standard, poiché possono essere simulate da queste ultime.

Esercizio 4

4. (9 punti) Un circuito Hamiltoniano in un grafo G è un ciclo che attraversa ogni vertice di G esattamente una volta. Stabilire se un grafo contiene un circuito Hamiltoniano è un problema NP-completo.

Considerate il seguente problema, che chiameremo HAM375: dato un grafo G con n vertici, trovare un ciclo che attraversa esattamente una volta $n-375$ vertici del grafo (ossia tutti i vertici di G *tranne* 375).

(a) Dimostra che HAM375 è un problema NP.
(b) Dimostra che HAM375 è NP-hard.

Soluzione

(a) Per dimostrare che HAM375 è un problema NP, dobbiamo mostrare che esiste un algoritmo deterministico polinomiale che verifica se una soluzione proposta di HAM375 è corretta. In altre parole, dobbiamo mostrare che se abbiamo un ciclo proposto che attraversa esattamente una volta n-375 vertici del grafo G, possiamo verificare in tempo polinomiale se questo ciclo è corretto.

Per verificare se un ciclo proposto è corretto, dobbiamo controllare che attraversi esattamente una volta n-375 vertici di G e che sia un ciclo, ovvero che il primo e l'ultimo vertice del ciclo siano lo stesso. Questo controllo richiede un tempo polinomiale rispetto alla dimensione del grafo, quindi HAM375 è un problema NP.

(b) Per dimostrare che HAM375 è NP-hard, riduciamo il problema HAM al problema HAM375. Supponiamo di avere un'istanza di HAM, ovvero un grafo G. Costruiamo un'istanza di HAM375 come segue: rimuoviamo 375 vertici qualsiasi dal grafo G, ottenendo un grafo G' con n-375 vertici. In questo modo, HAM375 richiede di trovare un ciclo che attraversa esattamente una volta n-375 vertici di G', ovvero tutti i vertici che non sono stati rimossi, il che corrisponde ad avere un ciclo Hamiltoniano in G.

Poiché la riduzione da HAM a HAM375 è polinomiale, segue che se esiste un algoritmo polinomiale per risolvere HAM375, allora esiste anche un algoritmo polinomiale per risolvere HAM. Tuttavia, poiché HAM è noto essere NP-completo, segue che HAM375 è NP-hard.

Concludiamo che HAM375 è un problema NP-completo.