Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, linea

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

(a

Immagine che contiene testo, Carattere, schermata, informazione

Descrizione generata automaticamente



Immagine che contiene testo, schermata, Carattere, documento

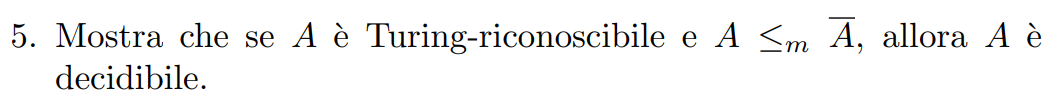
Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, informazione

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Per dimostrare che ogni linguaggio Turing-riconoscibile sull'alfabeto {0, 1, 2} può essere riconosciuto da una Turing machine con alfabeto ternario, possiamo procedere come segue:

Sia L un linguaggio Turing-riconoscibile sull'alfabeto {0, 1, 2}. Esiste una Turing machine M che riconosce L.

Costruiamo una Turing machine M' con alfabeto ternario Σ = {0, 1, \_} (dove "\_" rappresenta il blank) che riconosce L nel seguente modo:

1. M' simula M sul nastro, con la seguente codifica:

- 0 è codificato come 0

- 1 è codificato come 01

- 2 è codificato come 011

2. Quando M legge un simbolo a dal suo nastro, M' legge la sua codifica dal proprio nastro. In particolare:

- Se a = 0, M' legge 0

- Se a = 1, M' legge 01

- Se a = 2, M' legge 011

3. Quando M scrive un simbolo a sul suo nastro, M' scrive la sua codifica sul proprio nastro, sovrascrivendo tutti i simboli necessari.

4. M' simula le transizioni di stato e i movimenti della testina di M fedelmente.

5. Se M entra in uno stato di accettazione, anche M' entra in uno stato di accettazione. Se M entra in uno stato di rifiuto o non si ferma, lo stesso fa M'.

In questo modo, M' riconosce esattamente lo stesso linguaggio L di M, utilizzando solo l'alfabeto ternario {0, 1, \_}.

Infatti, per ogni stringa w in {0, 1, 2}\*, M accetta w se e solo se M' accetta la codifica di w in {0, 1, \_}\*.

Poiché L era un linguaggio Turing-riconoscibile arbitrario sull'alfabeto {0, 1, 2}, abbiamo dimostrato che ogni tale linguaggio può essere riconosciuto da una Turing machine con alfabeto ternario {0, 1, \_}.

In conclusione, abbiamo mostrato che le Turing machine con alfabeto ternario sono sufficienti per riconoscere tutti i linguaggi Turing-riconoscibili sull'alfabeto {0, 1, 2}, e quindi hanno la stessa potenza computazionale.

Immagine che contiene testo, Carattere, bianco, ricevuta

Descrizione generata automaticamente

1. Definire un linguaggio L che possa essere decisa da un 2-PDA.

2. Dimostrare che L non può essere deciso da alcun 1-PDA dimostrando che L non è libero dal contesto.

3. Dimostrare che L può essere deciso da una macchina di Turing.

Prendiamo come esempio la lingua L = {a^n b^n c^n | n >= 0}.

Passo 1: Dimostrare che L può essere deciso da un 2-PDA.

Possiamo costruire un 2-PDA che decide L come segue:

- Leggi le 'a' e spingile sulla prima pila.

- Leggi le 'b' e abbinale alle 'a' della prima pila. Spingi 'c' sulla seconda pila.

- Leggi le 'c' e abbinale alle 'c' della seconda pila.

- Accetta se tutti i simboli sono abbinati ed entrambe le pile sono vuote.

Passo 2: Dimostrare che L non può essere deciso da nessun 1-PDA.

Possiamo usare il lemma di pompaggio per i linguaggi context-free per mostrare che L non è context-free.

Si supponga che L sia privo di contesto. Sia p la lunghezza di pompaggio data dal lemma di pompaggio.

Si consideri la stringa s = a^p b^p c^p in L. Con il lemma di pompaggio, s può essere diviso in cinque parti, s = uvxyz, soddisfacendo le condizioni del lemma.

Tuttavia, il pompaggio di v e y risulterà in una stringa con un numero disuguale di 'a', 'b' e 'c', che non è in L. Questo contraddice il lemma di pompaggio.

Pertanto, L non è libero dal contesto e quindi non può essere deciso da alcun PDA 1.

Passo 3: Dimostrare che L può essere deciso da una macchina di Turing.

Possiamo costruire una macchina di Turing M che decide L come segue:

- Scansiona l'input e cancella una 'a', una 'b' e una 'c' alla volta fino a quando non è più possibile cancellare i simboli.

- Se l'input è esaurito e tutti i simboli sono stati cancellati, accettare. In caso contrario, rifiutare.

M decide L perché accetta tutte le stringhe in L e rifiuta tutte le stringhe non in L.

In conclusione, abbiamo dimostrato che il linguaggio L = {a^n b^n c^n | n >= 0} può essere deciso da un 2-PDA e da una macchina di Turing, ma non può essere deciso da nessun 1-PDA. Ciò dimostra che i 2-PDA sono più potenti degli 1-PDA in termini di classe di linguaggi che possono decidere, e che le macchine di Turing sono almeno altrettanto potenti dei 2-PDA.

Immagine che contiene testo, Carattere, schermata, linea

Descrizione generata automaticamente

Per dimostrare che il linguaggio L2 = {〈M, w〉 | M accetta la stringa ww^R} è indecidibile, possiamo procedere come segue:

Possiamo ridurre il problema della fermata (Halting problem) a L2. Il problema della fermata è noto per essere indecidibile.

Costruiamo una riduzione f che, dato un input 〈M, w〉 per il problema della fermata, genera una nuova macchina di Turing M' che funziona nel seguente modo:

- M' simula M sull'input w.

- Se M si ferma sull'input w, M' accetta la stringa ww^R (w concatenato con il suo reverse).

- Se M non si ferma su w, M' entra in un loop infinito su qualsiasi input.

Quindi, f restituisce 〈M', ww^R〉.

Se 〈M, w〉è un'istanza positiva del problema dell’arresto (cioè M si ferma su w), allora〈M', ww^R〉 è in L2 perché M' accetta ww^R.

Se 〈M, w〉è un'istanza negativa del problema della fermata (cioè M non si ferma su w), allora 〈M', ww^R〉non è in L2 perché M' non accetta alcun input.

Quindi, f è una riduzione dal problema della fermata a L2. Poiché il problema della fermata è indecidibile e riducibile a L2, anche L2 deve essere indecidibile.

In conclusione, abbiamo dimostrato che L2 è indecidibile riducendo il problema indecidibile della fermata ad esso. Ciò mostra che non può esistere un algoritmo generale per determinare se una macchina di Turing accetta una stringa nella forma ww^R.