Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, carta

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, documento

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

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, carta

Descrizione generata automaticamente

Immagine che contiene testo, ricevuta, Carattere, bianco

Descrizione generata automaticamente

Immagine che contiene testo, ricevuta, Carattere, bianco

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

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, bianco e nero

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

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

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, carta

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, bianco, ricevuta

Descrizione generata automaticamente

1. Dimostriamo che ogni linguaggio Turing-riconoscibile sull'alfabeto {0, 1, 2} può essere riconosciuto da una Turing machine con alfabeto ternario:

Sia M una TM che riconosce un linguaggio L sull'alfabeto {0, 1, 2}. Costruiamo una TM M' con alfabeto ternario {0, 1, 2, □} che simula M:

1. M' usa una codifica delle configurazioni di M sul suo nastro.
2. Per simulare una mossa di M, M' scansiona l'intera configurazione codificata, aggiornandola secondo la funzione di transizione di M.
3. M' accetta se e solo se M accetta.

La codifica delle configurazioni può essere fatta usando solo 0, 1, 2, □, ad esempio separando gli stati e i simboli con sequenze specifiche di questi simboli.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamenteQuesta costruzione mostra che M' può simulare M usando solo l'alfabeto ternario, quindi ogni linguaggio Turing-riconoscibile su {0, 1, 2} può essere riconosciuto da una TM con alfabeto ternario.

Sia M una TM che riconosce un linguaggio L. Costruiamo un automa deterministico a coda D che simula M:

1. D usa la coda per simulare il nastro di M. Inizialmente, D copia l'input nella coda.
2. D mantiene nella sua memoria finita lo stato corrente di M e la posizione della testina.
3. Per simulare una mossa di M: a. D scorre la coda fino alla posizione corrente della testina. b. Legge il simbolo, lo rimuove dalla coda e lo riscrive alla fine della coda. c. Aggiorna lo stato e la posizione della testina secondo la funzione di transizione di M.
4. Se M accetta, D accetta. Se M rifiuta, D rifiuta.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteQuesta costruzione mostra che D può simulare M, quindi ogni linguaggio Turing-riconoscibile può essere riconosciuto da un automa deterministico a coda.

Dimostriamo che qualsiasi macchina di Turing a testine multiple può essere simulata da una macchina di Turing deterministica a nastro singolo:

Sia M una TM a k testine. Costruiamo una TM S a singola testina che simula M:

1. S usa un nastro diviso in k+1 tracce:
   * La prima traccia contiene l'input originale.
   * Le altre k tracce contengono un marcatore per la posizione di ciascuna testina di M.
2. Per simulare una mossa di M: a. S scansiona l'intero nastro da sinistra a destra, memorizzando i k simboli letti dalle posizioni marcate. b. S calcola la nuova configurazione di M (nuovo stato, simboli da scrivere, movimenti delle testine). c. S scansiona di nuovo il nastro, aggiornando i simboli e le posizioni dei marcatori.
3. S accetta se e solo se M accetta.
4. Immagine che contiene testo, schermata, Carattere, linea

   Descrizione generata automaticamenteQuesta costruzione mostra che S può simulare M, quindi ogni TM a testine multiple può essere simulata da una TM a nastro singolo.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Idea: Useremo una macchina di Turing multinastro per la simulazione.

Sia M la macchina con stack di nastri. Costruiamo una macchina di Turing standard S con 3 nastri:

* Nastro 1: simula il nastro di lavoro di M
* Nastro 2: simula lo stack di M
* Nastro 3: tiene traccia del contenuto del nastro di M quando viene salvato nello stack

S simula M come segue: a) Per le operazioni normali, S usa il nastro 1 come M usa il suo nastro di lavoro. b) Per l'operazione Push:

* S copia il contenuto del nastro 1 sul nastro 3
* S sposta la testina del nastro 2 all'inizio e scrive un delimitatore
* S copia il contenuto del nastro 3 sul nastro 2 dopo il delimitatore c) Per l'operazione Pop:
* S verifica se il nastro 2 è vuoto. Se lo è, va nello stato di rifiuto.
* Altrimenti, S cerca l'ultimo delimitatore sul nastro 2
* S copia il contenuto dopo questo delimitatore sul nastro 1, sovrascrivendo il contenuto precedente
* S cancella questa porzione dal nastro 2

Poiché le macchine di Turing multinastro possono essere simulate da macchine di Turing a singolo nastro, questo dimostra che una macchina di Turing con stack di nastri può essere simulata da una macchina di Turing standard.

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Sia M = (Q, Γ, δ, q0, qaccept, qreject) una macchina di Turing ad albero binario. Costruiamo una macchina di Turing standard S che simula M:

S utilizza tre nastri:

* Nastro 1: contiene la codifica dell'albero binario
* Nastro 2: memorizza il percorso corrente nell'albero
* Nastro 3: simula il nastro di lavoro di M

Codifica dell'albero:

* Ogni nodo è rappresentato da una sequenza di L (sinistra) e R (destra) che indica il percorso dalla radice
* I nodi sono separati da #

Simulazione:

1. S inizializza il nastro 1 con l'input lungo il ramo sinistro
2. S inizializza il nastro 2 con ε (radice)
3. Per ogni passo di M: a. S legge il simbolo corrente dal nastro 1 usando il percorso nel nastro 2 b. S simula la transizione di M c. Se M si sposta a P (padre), S rimuove l'ultimo simbolo dal nastro 2 d. Se M si sposta a L o R, S aggiunge L o R al nastro 2 e. Se necessario, S espande l'albero sul nastro 1
4. S accetta se M accetta, rifiuta se M rifiuta

Questa costruzione mostra che ogni macchina di Turing ad albero binario può essere simulata da una macchina di Turing standard.

Immagine che contiene testo, schermata, Carattere, numero

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

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

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

Descrizione generata automaticamente

(a) Definizione formale della funzione di transizione di una CPTM: Una CPTM può essere definita come una tupla (Q, Σ, Γ, δ, q0, qaccept, qreject), dove:

* Q è l'insieme degli stati
* Σ è l'alfabeto di input
* Γ è l'alfabeto del nastro (include Σ e il simbolo blank)
* δ: Q × Γ → Q × Γ × {L, R, S, C, V, P} è la funzione di transizione, dove: L: muovi a sinistra R: muovi a destra S: rimani fermo C: seleziona l'inizio della porzione da copiare V: seleziona la fine della porzione da copiare P: incolla il contenuto copiato
* q0 è lo stato iniziale
* qaccept è lo stato di accettazione
* qreject è lo stato di rifiuto

(b) Dimostrazione che le CPTM riconoscono la classe dei linguaggi Turing-riconoscibili:

1. Ogni TM standard può essere simulata da una CPTM ignorando le operazioni C, V, P.
2. Per dimostrare che una CPTM può essere simulata da una TM standard:
   * Usa due nastri aggiuntivi: uno per memorizzare l'inizio della selezione, uno per il contenuto copiato.
   * Simula C memorizzando la posizione corrente sul primo nastro aggiuntivo.
   * Simula V copiando il contenuto selezionato sul secondo nastro aggiuntivo.
   * Simula P copiando il contenuto dal secondo nastro aggiuntivo al nastro principale. Questa simulazione mostra che le CPTM non sono più potenti delle TM standard.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

(a) Definizione formale della funzione di transizione di una JTM: Una JTM è definita come una tupla (Γ, Σ, δ, q, B, L, R, Y, N), dove:

* Γ è l'alfabeto del nastro (include Σ e i simboli speciali B, L, R, Y, N)
* Σ è l'alfabeto di input
* δ : Γ → Γ × {L, R} è la funzione di transizione
* q è l'unico stato della macchina
* B è il simbolo blank
* L, R sono i simboli per muovere la testina a sinistra o destra
* Y, N sono i simboli per accettare o rifiutare

(b) Dimostrazione che le JTM riconoscono la classe dei linguaggi Turing-riconoscibili:

1. Ogni JTM può essere simulata da una TM standard:
   * Usa un secondo nastro per memorizzare il simbolo che sarebbe stato scritto da una JTM
   * Simula ogni mossa della JTM usando questo nastro ausiliario
2. Per dimostrare che una TM standard può essere simulata da una JTM:
   * Codifica gli stati della TM sul nastro della JTM
   * Usa una codifica del nastro che alterna lo stato corrente e il contenuto del nastro
   * Implementa la funzione di transizione della TM usando la funzione di transizione della JTM

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamenteQuesta simulazione mostra che le JTM sono equivalenti alle TM standard in termini di potenza computazionale.

1. (a) Una UTM è una quintupla (Q, Σ, Γ, δ, q₀) dove:

* Q è un insieme finito di stati
* Σ è l'alfabeto di input che non contiene il simbolo blank \_
* Γ è l'alfabeto del nastro che contiene \_ e Σ
* δ : Q × Γ → Q × Γ × {L, R, UNDO} è la funzione di transizione, dove L, R indicano lo spostamento a sinistra/destra della testina, e UNDO l'annullamento dell'ultima operazione
* q₀ ∈ Q è lo stato iniziale

(b) Per dimostrare che le UTM riconoscono la classe dei linguaggi Turing-riconoscibili dobbiamo dimostrare due cose:

1. ogni linguaggio Turing-riconoscibile è riconosciuto da una UTM
2. ogni linguaggio riconosciuto da una UTM è Turing-riconoscibile
3. Banale: le TM a singolo nastro sono un caso particolare di UTM che non effettuano mai la mossa UNDO. Quindi ogni linguaggio Turing-riconoscibile è riconosciuto da una UTM.
4. Mostriamo come convertire una UTM M in una TM standard S equivalente a due nastri. S = "Su input w:
5. Simula M scrivendo ogni configurazione in sequenza sul nastro 1.
6. In caso di UNDO, torna indietro di due configurazioni sul nastro 1.
7. Quando M accetta/rifiuta, S accetta/rifiuta."

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteSiccome S riconosce lo stesso linguaggio di M e le TM a due nastri riconoscono i linguaggi Turing-riconoscibili, abbiamo dimostrato che ogni linguaggio riconosciuto da una UTM è Turing-riconoscibile.

1. (a) Una SRTM è una sestupla (Q, Σ, Γ, δ, q₀, q\_save) dove:

* Q è un insieme finito di stati
* Σ è l'alfabeto di input che non contiene il simbolo blank \_
* Γ è l'alfabeto del nastro che contiene \_ e Σ
* δ : Q × Γ → Q × Γ × {L, R, SAVE, RESTORE} è la funzione di transizione, dove L, R indicano lo spostamento a sinistra/destra della testina, SAVE salva la configurazione corrente, e RESTORE ripristina una configurazione salvata in precedenza
* q₀ ∈ Q è lo stato iniziale
* q\_save ∈ Q è lo stato "save e restore"

(b) Per dimostrare che le SRTM riconoscono la classe dei linguaggi Turing-riconoscibili usiamo lo stesso approccio dell'esercizio 1:

1. Banale: le TM a singolo nastro sono un caso particolare di SRTM che non effettuano mai le mosse SAVE e RESTORE. Quindi ogni linguaggio Turing-riconoscibile è riconosciuto da una SRTM.
2. Mostriamo come convertire una SRTM M in una TM standard S equivalente a più nastri. S = "Su input w:
3. Simula M sul nastro di lavoro
4. In caso di SAVE, copia la configurazione corrente su un nuovo nastro
5. In caso di RESTORE, cerca l'ultima configurazione salvata e ripristinala sul nastro di lavoro
6. Quando M accetta/rifiuta, S accetta/rifiuta."

Siccome S riconosce lo stesso linguaggio di M e le TM multi-nastro riconoscono i linguaggi Turing-riconoscibili, abbiamo dimostrato che ogni linguaggio riconosciuto da una SRTM è Turing-riconoscibile.