Lezione di Informatica Teorica: Varianti delle Macchine di Turing

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione alle Varianti delle Macchine di Turing

Le Macchine di Turing (MT) standard, pur essendo fondamentali per la teoria della computabilità, sono spesso complesse da programmare a causa della loro semplicità definitoria. Oggi esploreremo alcune varianti delle MT che, sebbene sembrino più potenti, si dimostreranno equivalenti alle MT standard in termini di potere computazionale, ma molto più facili da programmare.

1.1 Consigli per lo Studio

È utile estrapolare domande chiave dai concetti visti. Ad esempio:

- Cos'è un problema? Come si caratterizza?
- Problemi di ricerca vs. problemi di decisione: qual è il loro rapporto?
- Che ruolo hanno i linguaggi nei problemi di decisione?
- Cos'è una Macchina di Turing? Come è definita?
- Qual è la condizione di accettazione per una MT?
- Cosa significa che una MT decide un linguaggio vs. accetta un linguaggio? Qual è la differenza?

2 Macchine di Turing con Memoria nello Stato

Riprendiamo l'esempio del linguaggio $WW^R = \{ww^R \mid w \in \{0,1\}^*\}$. Nella MT standard per questo linguaggio, eravamo costretti a creare rami di computazione separati a seconda del primo carattere letto (0 o 1), perché la macchina doveva "ricordarsi" quel carattere. L'idea è: non sarebbe più semplice se la macchina potesse memorizzare direttamente il simbolo letto?

Definizione 1 (Macchina di Turing con Memoria nello Stato)**.** *Una Macchina di Turing con memoria nello stato è una MT che può memorizzare un simbolo dell'alfabeto del nastro all'interno del suo stato finito. Lo stato è quindi una coppia* $(q_i, \text{simbolo\_memorizzato})$*, dove* q_i *è lo stato del controllo finito e simbolo_memorizzato è il contenuto della "memoria interna".*

- *La macchina può memorizzare uno o più simboli (un numero fissato al momento della progettazione).*
- *La memoria non cresce durante l'esecuzione; è pre-determinata dal design.*

Esempio 1 (Linguaggio $L = \{w \mid w \in \{0,1\}^*$, inizia con α e prosegue solo con $\bar{\alpha}\}$, cioè $01^* \cup 10^*$)**.** *Vogliamo riconoscere stringhe binarie che iniziano con un simbolo e tutti gli altri sono diversi (es.* 0111*,* 1000*).*

- ***Stato Iniziale****:* $(q_0, \emptyset)$*, dove* $\emptyset$ *indica memoria vuota.*
- ***Lettura Primo Simbolo****:*
  - *Leggiamo un carattere* $\alpha \in \{0,1\}$*.*
  - *Lo lasciamo sul nastro.*
  - *Spostiamo la testina a destra (R).*

  - *Transitiamo allo stato (q_1, α), memorizzando α.*

- ***Ciclo di Verifica**:*
  - *Nello stato (q_1, α), leggiamo il simbolo $\bar{\alpha}$ (il complemento di α).*
  - *Lo lasciamo sul nastro.*
  - *Spostiamo la testina a destra (R).*
  - *Rimaniamo nello stato (q_1, α).*

- ***Accettazione**:*
  - *Nello stato (q_1, α), se leggiamo un simbolo `blank` ($\sqcup$).*
  - *Lasciamo $\sqcup$ sul nastro.*
  - *Spostiamo la testina a destra (R).*
  - *Transitiamo allo stato $(q_2, \emptyset)$ (o semplicemente q_2), e accettiamo.*

Questo approccio evita la necessità di duplicare gli stati per 0 e 1, rendendo la programmazione più compatta.

2.0.1 Potere Computazionale delle MT con Memoria nello Stato

Le MT con memoria nello stato *non* sono più potenti delle MT standard.

- Questa descrizione è un modo compatto per rappresentare una MT standard più grande.
- Gli stati di una MT con memoria nello stato sono di fatto ottenibili tramite il prodotto cartesiano degli stati del controllo finito e dei possibili simboli memorizzabili.
- Se lo stato (q_i, α) è uno stato, questo può essere semplicemente rinominato come $q_{i,\alpha}$ in una MT standard. Il numero di stati aumenta, ma la natura computazionale rimane la stessa.

3 Macchine di Turing Multitraccia

Le macchine multitraccia sono un altro trucchetto per semplificare la progettazione delle MT.

Definizione 2 (Macchina di Turing Multitraccia)**.** *Una Macchina di Turing multitraccia ha un singolo nastro, ma questo nastro è diviso orizzontalmente in più **tracce** (es. 2, 3, 7 tracce).*

- *Ogni cella del nastro può contenere un simbolo per ogni traccia.*
- *La macchina ha una **singola testina** di grandi dimensioni, capace di leggere e scrivere contemporaneamente su tutte le tracce di una data cella.*
- *Quando la testina si sposta (destra o sinistra), si sposta come un **monoblocco**, mantenendo la sua posizione relativa su tutte le tracce.*
- *La **funzione di transizione** è determinata dalla tupla di simboli letti su tutte le tracce nella posizione corrente. Può scrivere una tupla di nuovi simboli su tutte le tracce.*
- *Il numero di tracce è fissato al momento della progettazione e non può variare durante l'esecuzione.*

Esempio 2 (Linguaggio $L = \{w\#w \mid w \in \{a,b\}^+\}$)**.** *Simuliamo il riconoscimento di questo linguaggio con una MT multitraccia, usando due tracce: la prima per l'input e la seconda per i marcatori.*

- ***Stato Iniziale*** $(q_0, \emptyset)$.
- ***Passo 1: Lettura e Marcatura del Primo*** w
  - $(q_0, \emptyset)$*: Legge* $[\alpha, \sqcup]$ *(traccia 1:* α*, traccia 2: blank).*
  - *Scrive* $[\alpha, *]$ *(traccia 1:* α*, traccia 2: asterisco).*
  - *Sposta la testina a destra (R).*
  - *Transisce a* (q_1, α) *(memorizza* α*).*
  - *Questo processo si ripete per tutti i simboli di* w*.*
  - (q_1, α)*: Legge* $[\beta, \sqcup]$ *(qualsiasi simbolo* β *diverso da* α*).*
  - *Scrive* $[\beta, \sqcup]$.
  - *Sposta la testina a destra (R).*
  - *Rimane in* (q_1, α).
  - *Al cancelletto:*
    - (q_1, α)*: Legge* $[\#, \sqcup]$.
    - *Scrive* $[\#, \sqcup]$.
    - *Sposta la testina a destra (R).*
    - *Transisce a* (q_2, α).
- ***Passo 2: Verifica del Secondo*** w
  - (q_2, α)*: Salta i simboli già marcati (cercando il primo simbolo non marcato di* w^R*).*
  - *Legge* $[\beta, *]$ *(su traccia 1:* β*, su traccia 2: asterisco).*
  - *Scrive* $[\beta, *]$.
  - *Sposta la testina a destra (R).*
  - *Rimane in* (q_2, α).
  - *Quando trova il simbolo corrispondente a* α*:*
    - (q_2, α)*: Legge* $[\alpha, \sqcup]$.
    - *Scrive* $[\alpha, *]$.
    - *Sposta la testina a sinistra (L).*
    - *Transisce a* $(q_3, \emptyset)$ *(non serve più memorizzare* α*).*
- ***Passo 3: Ritorno all'Inizio del Primo*** w
  - $(q_3, \emptyset)$*: Salta i simboli marcati (su traccia 2).*
  - *Legge* $[\beta, *]$.
  - *Scrive* $[\beta, *]$.
  - *Sposta la testina a sinistra (L).*
  - *Rimane in* $(q_3, \emptyset)$.
  - *Al cancelletto:*

        - *$(q_3, \emptyset)$: Legge $[\#, \sqcup]$.*
        - *Scrive $[\#, \sqcup]$.*
        - *Sposta la testina a sinistra (L).*
        - *Transisce a $(q_4, \emptyset)$.*

- ***Passo 4: Preparazione per il Prossimo Simbolo***
    - *$(q_4, \emptyset)$: Salta i simboli non marcati (sulla traccia 2) per trovare il prossimo simbolo da marcare in w.*
    - *Legge $[\beta, \sqcup]$.*
    - *Scrive $[\beta, \sqcup]$.*
    - *Sposta la testina a sinistra (L).*
    - *Rimane in $(q_5, \emptyset)$. (Separato in q_5 e q_6 per distinguere tra simboli non marcati e marcati)*
    - *Quando trova il prossimo simbolo marcato:*
        - *$(q_5, \emptyset)$: Legge $[\beta, *]$.*
        - *Scrive $[\beta, *]$.*
        - *Sposta la testina a destra (R).*
        - *Torna a $(q_0, \emptyset)$ per ripetere il ciclo per il prossimo simbolo.*
    - ***Condizione di Accettazione (Fine Verifica)**:*
        - *Se, durante il ritorno da q_4, invece di leggere un simbolo non marcato, si legge un simbolo già marcato su traccia 2:*
            - *$(q_4, \emptyset)$: Legge $[\alpha, *]$.*
            - *Scrive $[\alpha, *]$.*
            - *Sposta la testina a destra (R).*
            - *Transisce a $(q_6, \emptyset)$. (Indica che tutti i simboli di w sono stati verificati)*
        - *Ora bisogna verificare che non ci sia più nulla dopo il cancelletto (cioè anche il secondo w è stato verificato).*
            - *$(q_6, \emptyset)$: Legge $[\#, \sqcup]$.*
            - *Scrive $[\#, \sqcup]$.*
            - *Sposta la testina a destra (R).*
            - *Transisce a $(q_7, \emptyset)$.*
        - *$(q_7, \emptyset)$: Salta i simboli marcati su traccia 2 (assicurandosi che tutto il secondo w sia marcato).*
            - *Legge $[\alpha, *]$.*
            - *Scrive $[\alpha, *]$.*
            - *Sposta la testina a destra (R).*
            - *Rimane in $(q_7, \emptyset)$.*
        - ***Accettazione Finale**:*
            - *$(q_7, \emptyset)$: Legge $[\sqcup, \sqcup]$ (blank su entrambe le tracce).*
            - *Scrive $[\sqcup, \sqcup]$.*
            - *Sposta la testina a destra (R).*
            - *Transisce a $(q_8, \emptyset)$ (Accetta).*

Questa macchina è più semplice da progettare rispetto alla versione a nastro singolo perché non si perde il contenuto originale dell'input.

3.0.1 Potere Computazionale delle MT Multitraccia

Le MT multitraccia *non* sono più potenti delle MT standard.

1. **Simulazione tramite Alfabeto Esteso**: Una MT multitraccia con K tracce può essere simulata da una MT standard (a singolo nastro e singola traccia) che usa un alfabeto esteso. Ogni simbolo del nuovo alfabeto è una tupla di K simboli, rappresentando la combinazione di simboli che apparirebbero su ogni traccia in una data cella. Ad esempio, se le tracce sono 2 e gli alfabeti $\{a, b\}$, $\{*, \sqcup\}$, il nuovo alfabeto può includere simboli come $(a, *)$, $(b, \sqcup)$, ecc. Questo aumenta la dimensione dell'alfabeto, ma non il potere computazionale.
2. **Simulazione tramite Spazio Aggiuntivo e Memoria nello Stato (meno comune per la dimostrazione di equivalenza):** Una MT standard può simulare una MT multitraccia prendendo l'input della MT multitraccia, inserendo delimitatori tra i simboli per creare "spazio" per le tracce aggiuntive. La MT standard userà poi la memoria nello stato per tenere traccia dei simboli che sarebbero sulle altre tracce e simulerà i movimenti della testina leggendo e scrivendo nei blocchi delimitati. Questo dimostra l'equivalenza ma è più complesso.

In sintesi, le MT multitraccia, sebbene sembrino più sofisticate, hanno lo stesso potere computazionale delle MT standard. Sono solo più facili da programmare.

4 Macchine di Turing Multinastro

Introduciamo il modello più conveniente per la programmazione: le Macchine di Turing Multinastro.

Definizione 3 (Macchina di Turing Multinastro)**.** *Una Macchina di Turing multinastro ha **più nastri completamente indipendenti** (es. 2, 3, 4 nastri).*

- *Ogni nastro ha la propria **testina indipendente**.*
- *Ogni testina può leggere, scrivere e muoversi **autonomamente** (destra (R), sinistra (L), o stare ferma (S)).*
- *L'input della macchina si trova sempre sul **primo nastro** all'avvio. Gli altri nastri (chiamati **nastri di lavoro** o* work tapes*) sono inizialmente vuoti.*
- *Il numero di nastri è fissato al momento della progettazione e non può cambiare durante l'esecuzione.*

Esempio 3 (Linguaggio $L = \{w \mid w \in \{0,1,2\}^*, \text{count}(0) = \text{count}(1) = \text{count}(2)\}$)**.** *Vogliamo decidere se il numero di '0', '1' e '2' in una stringa di input è uguale. Approccio con MT multinastro (4 nastri: 1 input, 3 di lavoro):*

1. ***Nastro 1 (Input)**: Contiene la stringa w.*
2. ***Nastro 2 (Zeros)**: Memorizzerà tutti gli '0'.*
3. ***Nastro 3 (Ones)**: Memorizzerà tutti gli '1'.*
4. ***Nastro 4 (Twos)**: Memorizzerà tutti i '2'.*

Strategia:

- ***Passo 1: Copia e Separazione (Stato q_0)**:*
  - *Si scansiona il Nastro 1 da sinistra a destra.*
  - *Se si legge '0' sul Nastro 1:*
    - *Mantiene '0' sul Nastro 1, muove testina 1 a destra (R).*
    - *Scrive '0' sul Nastro 2 (che era $\sqcup$), muove testina 2 a destra (R).*
  - *Se si legge '1' sul Nastro 1:*
    - *Mantiene '1' sul Nastro 1, muove testina 1 a destra (R).*
    - *Scrive '1' sul Nastro 3 (che era $\sqcup$), muove testina 3 a destra (R).*
  - *Se si legge '2' sul Nastro 1:*
    - *Mantiene '2' sul Nastro 1, muove testina 1 a destra (R).*
    - *Scrive '2' sul Nastro 4 (che era $\sqcup$), muove testina 4 a destra (R).*
  - *Questo continua finché il Nastro 1 non incontra $\sqcup$.*
- ***Passo 2: Riavvolgimento Testine di Lavoro (Transizione da q_0 a q_1)**:*
  - *Quando Nastro 1 legge $\sqcup$:*
    - *Nastro 1: $[\sqcup, \sqcup]$, testina 1 ferma (S).*
    - *Nastri 2, 3, 4 (che avranno la testina all'inizio della loro area blank): $[\sqcup, \sqcup]$, testine 2, 3, 4 muovono a sinistra (L).*
    - *Transisce allo stato q_1.*
- ***Passo 3: Confronto Conteggi (Stato q_1)**:*
  - *Nello stato q_1, le testine dei nastri 2, 3, 4 sono posizionate all'inizio dei simboli scritti.*
  - *Si muovono simultaneamente a sinistra (L), leggendo i simboli.*
  - *Se Nastro 2 legge '0', Nastro 3 legge '1', Nastro 4 legge '2':*
    - *Mantengono i simboli, muovono testine 2, 3, 4 a sinistra (L).*
    - *Rimangono in q_1.*
  - *Questo processo continua finché tutti i nastri incontrano il simbolo $\sqcup$ **contemporaneamente**.*
- ***Passo 4: Accettazione (Transizione da q_1 a q_2)**:*
  - *Quando Nastro 2, Nastro 3 e Nastro 4 leggono $\sqcup$ contemporaneamente:*
    - *Nastri 2, 3, 4: $[\sqcup, \sqcup]$, testine 2, 3, 4 ferme (S).*
    - *Transisce allo stato q_2 e accetta.*

Le MT multinastro sono molto convenienti perché permettono un controllo indipendente delle testine, rendendo algoritmi come questo (che altrimenti richiederebbero complessi vai-e-vieni sul nastro singolo) molto più intuitivi.

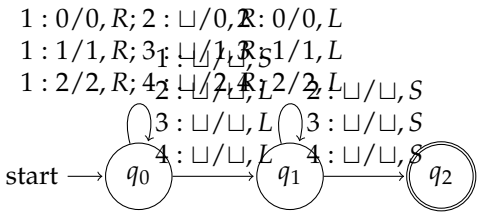

Figura 1: Diagramma di una Macchina di Turing Multinastro per il linguaggio $L = \{w \mid \text{count}(0) = \text{count}(1) = \text{count}(2)\}$

5 Equivalenza e Complessità Temporale

Come per le varianti precedenti, ci si chiede: le MT multinastro sono più potenti delle MT standard? La risposta è no, ma sono potenzialmente più veloci.

Definizione 4 (Tempo di Esecuzione di una MT)**.** *Il tempo di esecuzione di una Macchina di Turing su una certa stringa di input W è definito come il* ***numero di passi*** *(o il numero di configurazioni nella sua computazione) che la macchina compie prima di arrestarsi. Se la macchina non si arresta, il tempo di esecuzione è infinito. Questa definizione è molto precisa e permette un'analisi fine della complessità.*

Teorema 1 (Equivalenza MT Multinastro e MT Standard)**.** *Sia M una macchina di Turing multinastro. Allora esiste una macchina S a nastro singolo (e multitraccia) tale che il linguaggio riconosciuto da M è uguale al linguaggio riconosciuto da S.*

5.0.1 Dimostrazione (Sketch)

L'idea è simulare una MT multinastro M (con K nastri) tramite una MT S a nastro singolo ma multitraccia.

- **Architettura di** S: La macchina S avrà $2K$ tracce.
  - Le tracce dispari (es. $1, 3, 5, \ldots, 2K-1$) conterranno il contenuto dei K nastri di M.
  - Le tracce pari (es. $2, 4, 6, \ldots, 2K$) conterranno un marcatore speciale (es. '$*$') che indica la posizione della testina del nastro corrispondente in M.
  - Il resto della traccia pari sarà $\sqcup$.
- **Simulazione di un passo di** M **da parte di** S: Per simulare un singolo passo della macchina M, S deve eseguire i seguenti sottoprocessi:
  1. **Lettura dei Simboli**: S scansiona il suo nastro da sinistra a destra per trovare tutti i K marcatori delle testine. Quando trova un marcatore su una traccia pari, legge il simbolo corrispondente sulla traccia dispari immediatamente superiore e lo memorizza nello stato di S.
  2. **Decisione della Transizione**: Una volta letti tutti e K i simboli (e memorizzati nel suo stato), S sa quale transizione M avrebbe eseguito (basandosi sullo stato attuale di M e i K simboli letti). Questa transizione specifica quali simboli scrivere e in quale direzione muovere ciascuna testina.

3. **Aggiornamento del Nastro e Spostamento delle Testine**: S scansiona il suo nastro una seconda volta (es. da destra a sinistra), e in corrispondenza di ogni marcatore di testina:
   - Sovrascrive il simbolo sulla traccia dispari con il nuovo simbolo deciso nel passo precedente.
   - Sposta il marcatore '$*$' sulla traccia pari nella nuova posizione della testina (spostandolo a destra o a sinistra). Se il marcatore deve muoversi nella direzione opposta a quella della scansione attuale di S, ciò richiederà due mosse aggiuntive per il marcatore (sposta avanti, poi indietro e riprende la scansione).
4. **Ritorno alla Posizione Canonica**: S riporta la sua testina all'inizio del nastro (o a una posizione prefissata) per il prossimo ciclo di simulazione.

5.0.2 Analisi della Complessità Temporale della Simulazione

Sia M una MT multinastro che esegue N passi.

- Il tempo di esecuzione di M è N.
- Durante N passi, una testina di M non può allontanarsi più di N celle dalla sua posizione iniziale.
- Nella simulazione di S, le testine di M possono essere sparse sul nastro di S. Nella peggiore delle ipotesi, la testina di un nastro si trova a i posizioni a sinistra dell'inizio del nastro di S, e un'altra testina si trova a i posizioni a destra. La distanza massima tra due testine dopo i passi di M è $2i$ celle.
- Per simulare il i-esimo passo di M, S deve scansionare un segmento del suo nastro di lunghezza proporzionale alla massima distanza raggiunta dalle testine fino a quel momento, che è $O(i)$.
- Per ogni passo di M, S esegue due scansioni complete ($2 \times O(i)$) e potenzialmente $2K$ mosse aggiuntive (per i marcatori che vanno contro-scansione). Quindi, un passo di M costa S un tempo $O(i) + O(i) + O(K) = O(i + K)$.
- Il tempo totale per S per simulare N passi di M sarà la somma dei costi per ogni passo:

$$T_S(N) = \sum_{i=1}^{N} C \cdot (i + K) = C \cdot \sum_{i=1}^{N} i + C \cdot \sum_{i=1}^{N} K$$

$$T_S(N) = C \cdot \frac{N(N+1)}{2} + C \cdot NK = O(N^2 + NK) = O(N^2)$$

- Pertanto, una MT multinastro può essere simulata da una MT multitraccia (e quindi standard) con un **rallentamento quadratico**. Se M impiega N passi, S impiega $O(N^2)$ passi.

5.1 Conclusioni sul Potere Computazionale

- Le Macchine di Turing con memoria nello stato, le Macchine di Turing multitraccia e le Macchine di Turing multinastro sono tutte **computazionalmente equivalenti** alle Macchine di Turing standard. Non possono riconoscere linguaggi che una MT standard non possa riconoscere.

- Tuttavia, offrono notevoli vantaggi in termini di **facilità di programmazione** e possono garantire un'esecuzione **più efficiente** (con un rallentamento polinomiale, non esponenziale, nella simulazione).
- Per la loro praticità, le Macchine di Turing multinastro sono spesso il modello preferito nella pratica per studiare la complessità degli algoritmi.