Metodologia formale secondo Bresolin

🎯 OBIETTIVI TIPICI

• **Definire formalmente** varianti di Macchine di Turing • **Dimostrare equivalenze** tra varianti e TM standard • **Costruire simulazioni** bidirezionali • **Analizzare** proprietà computazionali

---

📚 DEFINIZIONI FONDAMENTALI

Macchina di Turing Standard

Una TM è una tupla **M = (Q, Σ, Γ, δ, q_0, qaccept, qreject)** dove:

- **Q**: insieme finito di stati
- **Σ**: alfabeto di input (non contiene ⊔)
- **Γ**: alfabeto del nastro (⊔ ∈ Γ, Σ ⊆ Γ)
- **δ: Q × Γ → Q × Γ × {L,R}**: funzione di transizione
- **q_0 ∈ Q**: stato iniziale
- **qaccept, qreject ∈ Q**: stati finali (qaccept ≠ qreject)

Equivalenza tra Modelli

Due modelli computazionali sono **equivalenti** se riconoscono la stessa classe di linguaggi.

Simulazione

Un modello A **simula** un modello B se ogni computazione di B può essere riprodotta da A.

🔧 METODOLOGIA PER EQUIVALENZE

SCHEMA STANDARD

Per dimostrare che una variante V è equivalente alle TM standard:

STEP 1: Definizione Formale della Variante

• **Specificare** la funzione di transizione modificata • **Descrivere** le operazioni aggiuntive/modificate • **Definire** il formato dell'input e configurazioni

STEP 2: Dimostrazione Bidirezionale

• **Direzione 1**: Ogni linguaggio riconosciuto da V è Turing-riconoscibile • **Direzione 2**: Ogni linguaggio Turing-riconoscibile è riconosciuto da V

STEP 3: Costruzioni di Simulazione

• **Per ogni direzione**, costruire esplicitamente la simulazione • **Describere algoritmi** a livello implementativo • **Verificare correttezza** delle simulazioni

---

📋 VARIANTI STANDARD E LORO SOLUZIONI

🔸 MACCHINE MULTINASTRO

Definizione

Una **k-nastro TM** ha:

- **k nastri** semi-infiniti
- **k testine** indipendenti
- **Input** sul nastro 1
- **Transizione**: $\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L,R\}^k$

Equivalenza con TM Standard

```
TEOREMA: Le TM multinastro riconoscono esattamente i linguaggi Turing-
riconoscibili.

DIMOSTRAZIONE:

DIREZIONE 1: Multinastro → Standard
Sia M una k-nastro TM. Costruiamo TM standard S:

S = "Su input w:
1. [INIZIALIZZAZIONE]
   Formatta nastro come: #w₁#w₂#...#wₖ#␣#␣#...
   Dove wᵢ rappresenta il contenuto del nastro i
   Marca posizioni testine con •simbolo
```

```
2. [SIMULAZIONE]
   Per simulare una mossa di M:
   a) Scansiona nastro per leggere simboli sotto testine virtuali
   b) Calcola prossima configurazione usando δ di M
   c) Aggiorna nastro scrivendo nuovi simboli e spostando •

3. [TERMINAZIONE]
   Quando M raggiunge stato finale, S termina analogamente"

DIREZIONE 2: Standard → Multinastro
Banale: TM standard è caso particolare di 1-nastro TM.

CONCLUSIONE: Equivalenza dimostrata. □
```

🔸 MACCHINE NON DETERMINISTICHE

Definizione

Una **NTM** ha:

- **Transizione**: δ: Q × Γ ⟶ P(Q × Γ × {L,R})
- **Accettazione**: accetta se almeno un cammino accetta
- **Linguaggio**: L(M) = {w | ∃ cammino computazionale accettante per w}

Equivalenza con TM Standard

```
TEOREMA: Le NTM riconoscono esattamente i linguaggi Turing-riconoscibili.

DIMOSTRAZIONE:

DIREZIONE 1: NTM → Deterministica
Sia N una NTM. Costruiamo TM deterministica S:

S = "Su input w:
1. [ENUMERAZIONE]
   Enumera sistematicamente tutti i possibili cammini computazionali di N su
w
   Usa strategia breadth-first per esplorare albero delle computazioni

2. [SIMULAZIONE]
   Per ogni cammino di lunghezza ≤ k:
   a) Simula k passi di N seguendo quel cammino
   b) Se cammino porta ad accettazione, ACCETTA
```

```
   c) Se tutti i cammini di lunghezza k sono stati esplorati, aumenta k

3. [TERMINAZIONE]
   Se esiste cammino accettante, S lo troverà eventualmente"

DIREZIONE 2: Deterministica → NTM
Banale: TM deterministica è caso particolare di NTM.

CONCLUSIONE: Equivalenza dimostrata. □
```

🔸 MACCHINE CON VARIANTI DI MOVIMENTO

Esempio: TM con Stay (S)

Funzione di transizione: δ: Q × Γ → Q × Γ × {L,R,S}

```
EQUIVALENZA:
S = "Su input w:
1. Simula TM con stay M usando TM standard
2. Per ogni mossa δ(q,a) = (r,b,S):
   • Scrivi b, sposta R, poi sposta L
   • Effetto netto: testina rimane sulla stessa cella
3. Altre mosse: simula normalmente"
```

Esempio: TM Bidimensionale

Nastro: griglia 2D infinita **Movimento**: δ: Q × Γ → Q × Γ × {↑,↓,←,→}

```
SIMULAZIONE:
S = "Su input w:
1. [RAPPRESENTAZIONE]
   Rappresenta griglia 2D come sequenza di righe:
   ##riga₁#riga₂#...#rigaₙ##
   Marca posizione testina con •

2. [MOVIMENTO ORIZZONTALE]
   → e ← : sposta • nella stessa riga

3. [MOVIMENTO VERTICALE]
   ↑ e ↓ : sposta • alla riga sopra/sotto stessa colonna
   Se riga non esiste, crea nuova riga vuota
```

```
4. [GESTIONE ESPANSIONE]
   Quando necessario, estendi rappresentazione aggiungendo righe/colonne"
```

🧩 COSTRUZIONI AVANZATE

🔹 Macchine con Operazioni Speciali

Esempio: TM con Accesso Casuale

```
PROBLEMA: TM può accedere direttamente alla posizione p dell'input

SOLUZIONE:
δ: Q × Γ × Γ → Q × Γ × {L,R} × {L,R} × {DIRECT}

SIMULAZIONE:
1. Nastro 1: input (sola lettura)
2. Nastro 2: lavoro
3. Nastro 3: puntatore (binario)

Per DIRECT:
• Leggi valore binario p dal nastro 3
• Vai alla posizione p del nastro 1
• Copia simbolo sul nastro 2
```

Esempio: TM con Stack

```
PROBLEMA: TM ha stack aggiuntivo (push, pop, top)

SOLUZIONE:
Simula stack usando nastro aggiuntivo:
• Push: sposta a destra, scrivi elemento
• Pop: leggi elemento, sposta a sinistra
• Top: leggi elemento senza spostare
```

🔹 Limitazioni e Vincoli

Esempio: TM con Alfabeto Binario

```
PROBLEMA: TM può usare solo {0,1,␣} sul nastro
```

```
SOLUZIONE:
1. [CODIFICA] Definisci codifica binaria per simboli originali
2. [SIMULAZIONE]
   • Converti input in codifica binaria
   • Simula operazioni sulla rappresentazione codificata
   • Decodifica quando necessario per decisioni
```

Esempio: TM Semi-infinito

```
PROBLEMA: Nastro ha inizio ma non fine (posizioni 0,1,2,...)

EQUIVALENZA: Già equivalente alle TM standard
NOTA: Definizione standard già assume nastro semi-infinito
```

⚙️ TECNICHE DI SIMULAZIONE AVANZATE

🔹 Gestione Multi-track

```
TECNICA: Usa "tracce" multiple su singolo nastro
APPLICAZIONE: Simulare k nastri con 1 nastro
IMPLEMENTAZIONE:
• Simbolo del nastro = (s₁, s₂, ..., sₖ)
• Ogni traccia rappresenta un nastro virtuale
• Scansione sequenziale per aggiornare tutte le tracce
```

🔹 Compressione Temporale

```
TECNICA: Ridurre numero di movimenti nella simulazione
APPLICAZIONE: Quando serve efficienza
IMPLEMENTAZIONE:
• Raggruppa operazioni multiple in singola scansione
• Usa rappresentazione compatta per configurazioni
```

🔹 Enumerazione Sistematica

```
TECNICA: Esplorare spazio computazionale sistematicamente
APPLICAZIONE: Simulazione di non-determinismo
IMPLEMENTAZIONE:
• Breadth-first search su albero computazioni
```

```
• Dovetailing per esplorare cammini infiniti
• Bound crescente su lunghezza cammini
```

⚠️ ERRORI COMUNI DA EVITARE

❌ Simulazioni Incomplete

• **Errore**: Non gestire tutti i casi della funzione di transizione • **Correzione**: Verificare sistematicamente ogni possibile transizione

❌ Rappresentazioni Ambigue

• **Errore**: Formato nastro non ben definito • **Correzione**: Specificare esattamente come informazione è codificata

❌ Terminazione Non Garantita

• **Errore**: Simulazione può non terminare anche se originale termina • **Correzione**: Dimostrare che simulazione preserva terminazione

❌ Equivalenza Unidirezionale

• **Errore**: Dimostrare solo una direzione dell'equivalenza • **Correzione**: Sempre dimostrare entrambe le direzioni

❌ Gestione Stati Finali

• **Errore**: Non preservare accettazione/rifiuto nella simulazione • **Correzione**: Mappare esplicitamente stati finali

---

📝 TEMPLATE GENERICO PER VARIANTI

```
PROBLEMA: Definire e dimostrare equivalenza di [Variante TM]

PARTE 1: DEFINIZIONE FORMALE
[Variante] è una tupla (Q, Σ, Γ, δ, q₀, qaccept, qreject) dove:
• Q, Σ, Γ, q₀, qaccept, qreject: come TM standard
• δ: [dominio modificato] → [codominio modificato]
• [Descrizione operazioni speciali]
```

```
PARTE 2: EQUIVALENZA
TEOREMA: [Variante] riconoscono esattamente i linguaggi Turing-
riconoscibili.

DIMOSTRAZIONE:

DIREZIONE 1: [Variante] → TM Standard
Sia V una [Variante]. Costruiamo TM standard S:

S = "Su input w:
1. [INIZIALIZZAZIONE]
   [Descrizione setup nastro per simulazione]

2. [SIMULAZIONE]
   Per simulare mossa di V:
   [Algoritmo dettagliato per ogni tipo di mossa]

3. [TERMINAZIONE]
   [Gestione stati finali]"

CORRETTEZZA: [Argomento che S simula correttamente V]

DIREZIONE 2: TM Standard → [Variante]
[Costruzione o argomento che TM standard sono caso particolare]

CONCLUSIONE: Equivalenza dimostrata. □

PARTE 3: ANALISI COMPLESSITÀ (opzionale)
[Overhead della simulazione]
```

🎯 CHECKLIST FINALE

Per ogni variante TM:

- [ ] Definizione formale completa (tutti i componenti)
- [ ] Funzione di transizione specificata precisamente
- [ ] Semantica operazionale chiara
- [ ] Entrambe le direzioni dell'equivalenza dimostrate
- [ ] Simulazioni costruttive e dettagliate
- [ ] Correttezza delle simulazioni argomentata

- [ ] Gestione corretta di stati finali
- [ ] Preservazione di terminazione/non-terminazione

Varianti Standard da Conoscere:

- [ ] TM Multinastro
- [ ] TM Non Deterministiche
- [ ] TM Bidimensionali
- [ ] TM con Stay
- [ ] TM con Alfabeto Limitato
- [ ] TM con Accesso Casuale
- [ ] TM con Stack/Queue
- [ ] TM Offline (input read-only)

Tecniche di Simulazione:

- [ ] Multi-track per simulare nastri multipli
- [ ] Enumerazione per non-determinismo
- [ ] Codifica per alfabeti limitati
- [ ] Rappresentazione gerarchica per strutture 2D
- [ ] Marcatura posizioni per testine multiple