Logica e Modelli Computazionali

Computabilità

Marco Console

Ingegneria Informatica e Automatica (Sapienza, Università di Roma)

Un Po’ di Storia

- Il modello della Macchina di Turing è stato ideato da **Alan Turing**, un matematico britannico
  - Una breve biografia di A. Turing è presente su Wikipedia (https://en.wikipedia.org/wiki/Alan_Turing)
  - Ne consiglio la lettura per avere una prospettiva delle sua vita e delle tragiche vicende personali
  - Il film "The Imitation Game" è tratto dal libro Alan Turing: The Enigma (Andrew Hodges)
- **Highlight 1**. A. Turing ha avuto un ruolo importante nell’intelligence della 2 Guerra Mondiale
  - È una degli attori principali nella decodifica del codice Enigma utilizzato dalla Germania Nazista per comunicare in maniera sicura informazioni militari sensibili
  - Ha ideato **Bombe**, la macchina elettro-meccanica utilizzata nella decodifica!
  - Secondo gli esperti il suo lavoro ha ridotto la durata della guerra di due anni!!
- **Highlight 2**. Nel 1952 viene condannato per "Gross Indecency" in UK a causa della sua omosessualità
  - Perde il lavoro a causa della revoca della Security Clearance
  - Viene costretto alla castrazione chimica
- **Highlight 3**. Nel 1954 si toglie la vita
  - In circostanze non molto chiare ma probabilmente a causa dello stato di depressione indotto dalla condanna
- **Highlight 4**. Il suo "reato" viene perdonato dalla corona britannica solo nel 2013
  - …

Teoria della Computabilità

- Il modello della Macchina di Turing nasce per rispondere a una domanda molto naturale:

Dato un problema P, è possibile definire una procedura automatica che risolva P?

- Fra i primi problemi ad essere studiati sotto questo aspetto c'è la tautologia al primo ordine
  - David Hilbert e Wilhelm Ackermann nel1928 definiscono ***Entscheidungsproblem*** (**problema della decisione**): esiste una **procedura** che, data in input una **formula della logica del primo ordine**, mediante un numero **finito di operazioni** sia in grado di stabilire se tale formula è o meno una tautologia

- Il problema così definito non ha chiare basi matematiche
  - Che cosa significa **procedura**? Quali istruzioni questa procedura può eseguire?
    - Se qualunque istruzione è ammesso, possiamo assumere un passo "controlla se φ è una tautologia" ☺
  - Su quale macchina la eseguiamo?

Teoria della Computabilità – II

- Nel 1936, **Alonzo Church** e **Alan Turing** (indipendetemente) rispondono **negativamente** all'*Entscheidungsproblem*

- Per fare ciò, ideano due diversi **modelli di computazione**,
  - Alonzo Church. **λ-calculus**
  - Alan Turing. **Macchina di Turing**,

- Tali modelli vogliono catturare tutte le funzioni matematiche "**effettivamente computabili**"
  - Tale nozione non ha una definizione matematica rigorosa!

- Le due prove hanno struttura simile
  - **Premessa**. Una funzione è effettivamente computabile se e solo se è **λ-computabile** (Church) **T-computabile** (Turing)
  - **Dimostrazione**. Entscheidungsproblem non è **λ-computabile** (Church) **T-computabile** (Turing)

Teoria della Computabilità – III

- Successivamente **Turing, Church e Stephen C. Kleene** dimostrano che λ-calculus e Macchine di Turing hanno lo stesso potere computazionale

- **Teorema**. Una funzione è T-computabile se e solo se è λ-computabile

- Tale risultato ha portato alla definizione della **tesi di Church-Turing**:

Una funzione è effettivamente computabile se e solo se è T-computabile

- La tesi di Church e Turing è stata ulteriormente rafforzata dal fatto che tutti i modelli di computazione che conosciamo sono riducibili alle Macchine di Turing
  - Esempio 1. La **macchina RAM** che simula una macchina a registri che esegue **Assembly**
  - Esempio2. La **macchina di Turing quantistica** che simula i computer quantistici

Computabilità

- Alla luce di quanto abbiamo detto, in quanto segue daremo una prova del seguente

- **Teorema 1.** Esiste un linguaggio $\mathcal{L}$ per cui non esiste una MT M tale che $L(M) = \mathcal{L}$

- **Teorema 1** (sotto la tesi di Church e Turing) ci dice che esiste almeno una funzione che non può essere effettivamente computata
  - La funzione $f: \Sigma^* \to \{0,1\}$ tale che $\mathrm{f}(s) = 1$ se e solo se $s \in \mathcal{L}$

- In realtà, costruiremo strumenti che ci consentono di provare l'esistenza di molti tali problemi

Macchine Multi-Nastro

Macchina di Turing Multi-Nastro – Intuizione

- Il modello delle macchine di Turing può essere esteso senza alterarne il potere computazionale.
- Per semplificare le prove che seguono, utilizzeremo le macchine multi-nastro
  - Probabilmente una delle più semplici estensioni delle macchine di Turing
- **Intuizione**. Tali macchine hanno accesso ad un insieme finito di nastri e testine indipendenti
  - La funzione di transizione determina il comportamento di ogni testina
  - In questo modo la macchina può confrontate il contenuto di diverse aree di memoria contemporaneamente
- **Intuizione.** La capacità di riconoscere e decidere linguaggi delle macchine multi-nastro rimane invariata

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	b	b	c	c	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

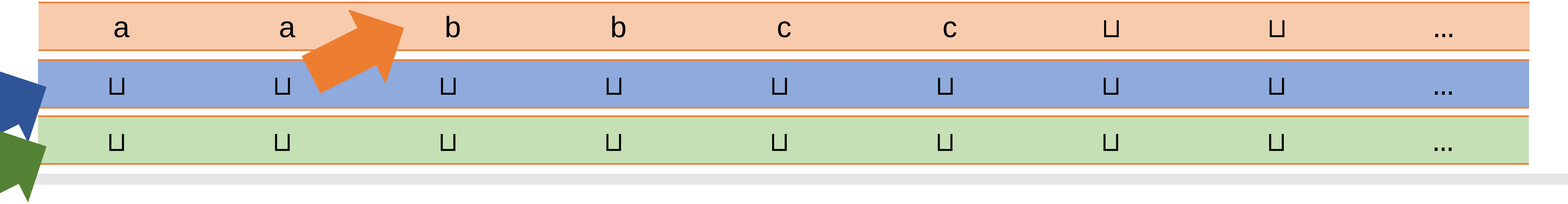
a	a	b	b	c	c	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)



Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	⊔	b	c	c	⊔	⊔	...
b	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	⊔	⊔	c	c	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	⊔	⊔	⊔	c	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
c	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	⊔	⊔	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
c	c	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
   1. Il primo nastro contenga solo a
   2. Il secondo nastro contenga solo b
   3. Il terzo nastro contenga solo c
   4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	⊔	⊔	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
c	c	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	⊔	⊔	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
c	c	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
   1. Il primo nastro contenga solo a
   2. Il secondo nastro contenga solo b
   3. Il terzo nastro contenga solo c
   4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...
b	b	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...
c	c	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

Stringa Accettata!

a	a	⊔	⊔	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
c	c	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
   1. Il primo nastro contenga solo a
   2. Il secondo nastro contenga solo b
   3. Il terzo nastro contenga solo c
   4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	c	c	b	b	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

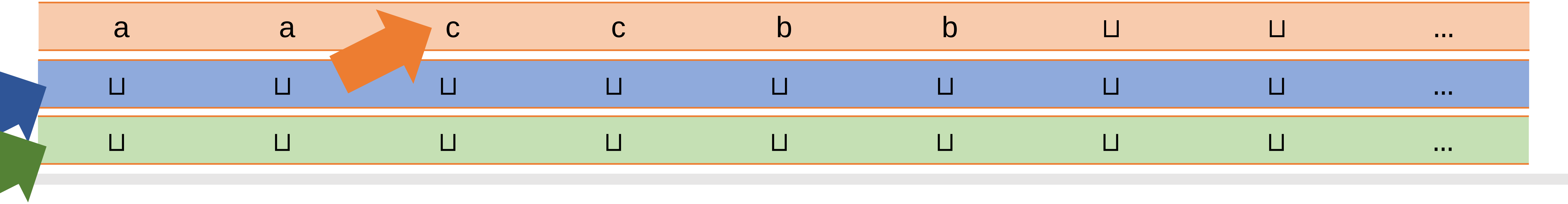
a	a	c	c	b	b	$\sqcup$	$\sqcup$	...
$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...
$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)



Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima $\sqcup$ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui $\sqcup$ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	c	c	b	b	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	c	c	b	b	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	c	c	⊔	b	⊔	⊔	...
b	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	c	c	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
    1. Il primo nastro contenga solo a
    2. Il secondo nastro contenga solo b
    3. Il terzo nastro contenga solo c
    4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

a	a	c	c	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchine di Turing Multi-Nastro – Esempio di Computazione

Algoritmo per riconoscere $L = \{a^m b^m c^m \mid m \geq 1\}$

- **Supponiamo di avere 3 nastri distinti e input σ**

1. Sposta la sottostringa di σ dalla prima b alla prima c (non inclusa) nel secondo nastro
2. Sposta la sottostringa di σ dalla prima c alla prima ⊔ (non inclusa) nel terzo nastro
3. Confronta il contenuto dei tre nastri e verifica che
   1. Il primo nastro contenga solo a
   2. Il secondo nastro contenga solo b
   3. Il terzo nastro contenga solo c
   4. La prima configurazione in cui ⊔ compare in uno dei nastri segna la fine delle tre stringhe (stessa cardinalità)

Stringa Rifiutata!

a	a	c	c	⊔	⊔	⊔	⊔	...
b	b	⊔	⊔	⊔	⊔	⊔	⊔	...
⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	...

Macchina di Turing Multi-Nastro – Definizione Formale

Definizione. Una **macchina di Turing con $k \geq 1$ nastri** (**k-TM**) M è una 7-upla $M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle$ tale che:

- Σ è un insieme finito di simboli detto, **insieme dei simboli di input** che **non include** il **simbolo blank** ⊔
- Γ con $\Sigma \subseteq \Gamma$ è un insieme finito di simboli detto **insieme dei simboli del nastro**, che **include** il **simbolo blank** ⊔
- Q è un insieme finito e non vuoto di **stati** con:
- $q_0 \in Q$ è lo **stato iniziale**
- $q_{yes} \in Q$ è lo **stato accettante**
- $q_{no} \in Q$ è lo **stato rifiutante**
- δ è la **funzione di transizione**; ovvero, una funzione totale definita come segue

$$\delta: Q \times \Sigma^k \rightarrow \mathrm{Q} \times (\Gamma \times \{\leftarrow, \rightarrow, -\})^k$$

Macchina di Turing Multi-Nastro – Informale

- **Intuizione 1**. Gli stati di una Macchina di Turing multi-nastro operano come quelli standard
  - Lo **stato accettante** termina la computazione della macchina e **accetta l'input**
  - Lo **stato rifiutante** termina la computazione della macchina e **rifiuta l'input**

- **Intuizione 2**. La **funzione di transizione,** invece, riceve in input lo coppia $(q, \sigma) \in Q \times \Sigma^{\mathrm{k}}$ composta dallo stato interno e k simboli correntemente letti da k "testine" su k nastri paralleli e restituisce
  - il prossimo stato interno della macchina (uno solo)
  - k simboli che sovrascrivono i correnti k (letti dai nastri paralleli)
  - k spostamenti delle testine (effettuati sui nastri paralleli)

- **Intuizione 3**. Il **prossimo passo** di una Macchina di Turing con k nastri è determinato dallo stato interno, dal contenuto di k nastri distinti e dalla posizione di k testine distinte
  - Da queste informazioni possiamo definire la sequenza eseguiti dalla macchina (**computazione**)

Macchina di Turing Multi-Nastro – Configurazioni

- Sia $M = < \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} >$ una **macchina di Turing con k nastri**

- **Definizione**. Una **configurazione** di M è una $(2k+1)$-upla $C = (q,\ \sigma_1, \tau_1, \sigma_2, \tau_2, \dots, \sigma_k, \tau_k)$ t.c.:
  - σ_i, per ogni $i = 1, \dots, k$, è una **stringa sull'alfabeto** Γ (nastro a sinistra della i-esima testina)
  - $\tau_i = a_i \tau_i{}'$, per ogni $i = 1, \dots, k$, è una **stringa sull'alfabeto** Γ (nastro a destra della i-esima testina)
  - $q \in Q$ è uno **stato** di M (stato corrente della configurazione)

- **Definizione**. Una **configurazione** $C = (\sigma, \mathrm{q}, \tau)$ di M è detta:
  - **Accettante** (**finale**) se $q = q_{yes}$; **Rifiutante** (**finale**) se $q = q_{no}$;
  - **Iniziale** se $q = q_0$, $\sigma_i = \epsilon$ e $\tau_i \in \Sigma^*$ per ogni $i = 1, \dots, k$

Macchina di Turing Multi-Nastro – Configurazioni

- **Esempio**. Sia $\mathrm{M} =< \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} >$ con $\Sigma = \{a, b\}, \Gamma = \Sigma \cup \{\sqcup\}, Q = \{q_0, q_1, q_{yes}, q_{no}\}$ una **2-TM**

  - $(q_0, \epsilon, aa, \epsilon, bb)$ è una configurazione **Iniziale** di M
  - $(q_{yes}, abab, abb \sqcup, \sqcup, b)$ è una configurazione **Accettante** di M
  - $(q_{no}, aab, a \sqcup b, a, \sqcup b \sqcup)$ è una configurazione **Rifiutante** di M
  - $(q_1, aab, a \sqcup b, a, \sqcup b \sqcup)$ è una configurazione di M (non iniziale, non finale)

Macchina di Turing Multi-Nastro – Esecuzioni

- Sia $M = < \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} >$ una **k-TM**

- **Definizione**. Una **configurazione** C **genera** una **configurazione** D in M ($C \Rightarrow_M D$) se
  - $C = (q,\ \sigma_1, \tau_1, \sigma_2, \tau_2, \dots, \sigma_k, \tau_k)$
  - $D = (q',\ \sigma'_1, \tau'_1, \sigma'_2, \tau'_2, \dots, \sigma'_k, \tau'_k)$
  - $\delta(q,\ \sigma_1, \tau_1, \sigma_2, \tau_2, \dots, \sigma_k, \tau_k) = (q', x_1, m_1, \dots, x_k, m_k)$
  - Ogni configurazione (σ_i, q, τ_i), per $i = 1, \dots, k$, genera la configurazione (σ'_i, q', τ'_i) applicando la transizione (q', x_i, m_i) **con le regole della Macchina di Turing standard**
    - La macchina manipola i k nastri contemporaneamente applicando k funzioni di transizioni
    - Tutte le k testine hanno vita propria e possono operare in autonomia

Macchina di Turing Multi-Nastro – Esecuzioni – Esempio

- Sia $\mathrm{M} =< \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} >$ con $\Sigma = \{a, b\}, \Gamma = \Sigma \cup \{\sqcup\},\ Q = \{q_0, q_{yes}, q_{no}\}$
  - $\delta(q_0, a, a) = (q_0, a, \rightarrow, a, \rightarrow);\ \delta(q_0, b, b) = (q_0, b, \rightarrow, b, \rightarrow)$
  - $\delta(q_0, a, b) = (q_{no}, a, -, b, -);\ \delta(q_0, a, \sqcup) = (q_{no}, a, -, \sqcup, -)$
  - $\delta(q_0, b, a) = (q_{no}, b, -, a, -);\ \delta(q_0, b, \sqcup) = (q_{no}, b, -, \sqcup\ -)$
  - $\delta(q_0, \sqcup, a) = (q_{no}, \sqcup, -, a, -);\ \delta(q_0, \sqcup, b) = (q_{no}, \sqcup, -, b, -)$
  - $\delta(q_0, \sqcup, \sqcup) = (q_{yes}, \sqcup, -)$

- **Esempio.** La **configurazione iniziale** $\mathrm{C}_1 = (q_0, \epsilon, aa, \epsilon, aa)$ genera la configurazione $\mathrm{C}_2 = (q_0, a, a, a, a)$
- **Esempio.** La configurazione $\mathrm{C}_2 = (q_0, a, a, a, a)$ genera la configurazione $\mathrm{C}_3 = (q_0, aa, \epsilon, aa, \epsilon)$
- **Esempio.** La configurazione C_3genera la **configurazione accettante** $\mathrm{C}_4 = \left(q_{yes}, aa, \epsilon, aa, \epsilon\right)$

- **Esempio.** La **configurazione iniziale** $\mathrm{D}_1 = (q_0, \epsilon, bb, \epsilon, ba)$ genera la configurazione $\mathrm{D}_2 = (q_0, b, b, b, a)$
- **Esempio.** La configurazione D_2 genera la **configurazione rifiutante** $\mathrm{D}_3 = (q_{no}, b, b, b, a)$

Macchina di Turing Multi-Nastro – Linguaggio Riconosciuti

- **Definizione**. La k-TM M **accetta\rifiuta l'input $\sigma \in \Sigma^*$** se esiste una **sequenza finita** di configurazioni $C_1, C_2, \ldots, C_n$ di M tale che le seguenti proprietà sono soddisfatte
  - $C_1 = (q_0, \epsilon, \sigma, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon, \epsilon \ldots, \epsilon, \epsilon, \epsilon, \epsilon)$ (l'input è sul primo nastro)
  - C_n è una configurazione **accettante\rifiutante**
  - $C_i \Rightarrow_M C_{i+1}$ per ogni $i = 1, \ldots, n-1$

- **Definizione**. L'insieme $L(M)$ delle stringhe **che M riconosce** è detto **il linguaggio riconosciuto da M**

- **Definizione**. La k-TM M è **terminante** se per ogni stringa $\sigma \in \Sigma$ M **accetta** o **rifiuta** σ

- **Teorema**. Sia $\mathcal{L}$ un linguaggio di stringhe.
  1. $\mathcal{L}$ è **riconosciuto da un k-TM** se e solo se $\mathcal{L}$ è **Turing Riconoscibile**
  2. $\mathcal{L}$ è **riconosciuto da un k-TM terminante** se e solo se $\mathcal{L}$ è **Turing Decidibile**

La macchina di Turing Universale

La Macchina di Turing Universale – Intuizione

- La caratteristica principale di un moderno calcolatore è quella di seguire programmi
  - Solo calcolatori molto semplici sono vincolati al codice del loro "firmware"
  - La Macchina di Turing ha l'esatta stessa funzionalità!

- Una **Macchina di Turing Universale** U è una Macchina di Turing che, data la ***descrizione*** di una Macchina di Turing M e un input x per M, allora

$$\boldsymbol{(M, x) \in L(U)} \textbf{ se e solo se } \boldsymbol{x \in M(L)}$$

- **Intuizione**. L'obiettivo di U è quello di "simulare" il comportamento di M con input x

Macchina di Turing Universale – Encoding dell'Input

- Per definire una **Macchina di Turing Universale** U partiamo **dall'Encoding dell'input**
  - Encoding: funzione che definisce rappresentazioni di oggetti matematici come stringhe

- **Definizione 1.** Sia $\mathrm{b}(n)$ per $n \in \mathbb{N}$ l'Encoding di n in una stringa in $\{0,1,\#\}^*$ che rappresenta n in binario seguito da $\#$

- **Definizione 2.** Sia $M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle$ e definiamo (senza perdita di generalità)
  - $\Gamma = \{0, \dots, \mathrm{n_g}\} \subseteq \mathbb{N}$ (i simboli sono solo numeri, anche $\sqcup$ che possiamo assumere essere 0);
  - $\mathrm{Q} = \{0, \dots, \mathrm{n_q}\} \subseteq \mathbb{N}$ (gli stati sono solo numeri, assumiamo sempre che che $q_0 = 0, q_{yes} = 1, q_{no} = 2$)

- **Definizione 3**. Sia $\mathrm{q} \in Q$ e $c \in \Gamma$ e $\delta(q, c) = (q', c', m)$. L'Encoding $e(q, c)$ per M è la stringa
$$\big(b(q), b(c)\big)\big(b(q'), b(c'), m\big)$$

- **Definizione 4**. Sia $c_1 c_2 \dots c_k \in \Sigma^*$. L'Encoding $e(M, \sigma)$ di M e σ è la stringa $e(M); e(\sigma)$ dove
$$e(M) = e(0,0); e(0,1); \dots; e\big(n_q, n_g\big)$$
$$e(\sigma) = e(c_1) e(c_2) \dots e(c_k)$$

Macchina di Turing Universale – Encoding – Esempio

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 122}$

$\boldsymbol{(q,x)}$	$\boldsymbol{\delta(q,x)}$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

- L'Encoding $b(M, \sigma)$ è il seguente

$$\boldsymbol{(0\#,0\#)(0\#,0\#,\rightarrow);(0\#,1\#)(10\#,1\#,-);(0\#,10\#)(1\#,10\#,-);1\#10\#10\#}$$

Macchina di Turing Universale – Comportamento

- Definiamo una **Macchina di Turing Universale** U che utilizza l'encoding definito in precedenza

- La macchina U è una Macchina di Turing a multi-nastro. Su input M, σ la macchina U
  - **Nastro 1**. Riceve l'input e lo utilizzato per tenere traccia della funzione di transizione di M
  - **Nastro 2**. Lo utilizza come nastro di computazione per la simulazione
  - **Nastro 3**. Lo utilizza per tenere traccia dello stato di M corrente durante la simulazione

- La macchina U si comporta come segue su input $e(M); e(\sigma)$
  1. Copia $e(\sigma)$ (Encoding di σ) su **Nastro 2** e riporta **Testina 2** alla prima cella
  2. Copia $0\#$ su **Nastro 3** (Encoding dello stato iniziale di M) e riporta **Testina 3** alla prima cella
  3. Cerca in **Nastro 1** la transizione per il simbolo corrente in **Nastro 2** lo stato in **Nastro 3**
  4. Applica tale transizione a **Nastro 2** (simbolo e movimento testina) e **Nastro 3** (aggiorna stato)
  5. Se **Nastro 3** contiene $1\#$ (rappresentazione di q_{yes}) allora termina la computazione e **Accetta**
  6. Se **Nastro 3** contiene $10\#$ (rappresentazione di q_{no}) allora termina la computazione e **Rifiuta**
  7. **Altrimenti torna a 2**

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\mathbf{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -); 0\#0\#1\#10\# \sqcup\sqcup \cdots}$

Nastro 2: $\sqcup\sqcup \cdots$

Nastro 3: $\mathbf{0\#} \sqcup \cdots$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

$\boldsymbol{(q,x)}$	$\boldsymbol{\delta(q,x)}$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#,0\#)(0\#,0\#,\rightarrow);(0\#,1\#)(10\#,1\#,-);(0\#,10\#)(1\#,10\#,-);\sqcup\sqcup\cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#\sqcup\sqcup\cdots}$

Nastro 3: $\boldsymbol{0\#\sqcup\cdots}$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0, \rightarrow)$
$(0,1)$	$(2,1, -)$
$(0,2)$	$(1,2, -)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -);} \sqcup\sqcup \cdots$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#} \sqcup\sqcup \cdots$

Nastro 3: $\boldsymbol{0\#} \sqcup \cdots$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

$\boldsymbol{(q, x)}$	$\boldsymbol{\delta(q, x)}$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -); \sqcup\sqcup \cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\# \sqcup\sqcup \cdots}$

Nastro 3: $\boldsymbol{0\# \sqcup \cdots}$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#,0\#)(0\#,0\#,\rightarrow);(0\#,1\#)(10\#,1\#,-);(0\#,10\#)(1\#,10\#,-);\sqcup\sqcup\cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#\sqcup\sqcup\cdots}$

Nastro 3: $\boldsymbol{0\#\sqcup\cdots}$

Lo stato ragiunto non è finale. Prosegui.

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q,x)	$\delta(q,x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#,0\#)(0\#,0\#,\rightarrow);(0\#,1\#)(10\#,1\#,-);(0\#,10\#)(1\#,10\#,-);\sqcup\sqcup\cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#\sqcup\sqcup\cdots}$

Nastro 3: $\boldsymbol{0\#\sqcup\cdots}$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -);} \sqcup\sqcup \cdots$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#} \sqcup\sqcup \cdots$

Nastro 3: $\boldsymbol{0\#} \sqcup \cdots$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -); \sqcup\sqcup \cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\# \sqcup\sqcup \cdots}$

Nastro 3: $\boldsymbol{0\# \sqcup \cdots}$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -); \sqcup\sqcup \cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\# \sqcup\sqcup \cdots}$

Nastro 3: $\boldsymbol{0\# \sqcup \cdots}$

Lo stato ragiunto non è finale. Prosegui.

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -); \sqcup\sqcup \cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\# \sqcup\sqcup \cdots}$

Nastro 3: $\boldsymbol{0\# \sqcup \cdots}$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -);} \sqcup\sqcup \cdots$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#} \sqcup\sqcup \cdots$

Nastro 3: $\boldsymbol{0\#} \sqcup \cdots$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q, x)	$\delta(q, x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#, 0\#)(0\#, 0\#, \rightarrow); (0\#, 1\#)(10\#, 1\#, -); (0\#, 10\#)(1\#, 10\#, -); \sqcup\sqcup \cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\# \sqcup\sqcup \cdots}$

Nastro 3:$\boldsymbol{10\# \sqcup \cdots}$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q,x)	$\delta(q,x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $(\mathbf{0\#, 0\#})(\mathbf{0\#, 0\#}, \rightarrow); (\mathbf{0\#, 1\#})(\mathbf{10\#, 1\#}, -); (\mathbf{0\#, 10\#})(\mathbf{1\#, 10\#}, -); \sqcup\sqcup \cdots$

Nastro 2: $\mathbf{0\#0\#1\#10\#} \sqcup\sqcup \cdots$

Nastro 3: $\mathbf{10\#} \sqcup \cdots$

Macchina di Turing Universale – Esempio di Computazione

- Presentiamo l'Encoding per l'input M, σ
- $\boldsymbol{M = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{yes}, q_{no} \rangle}$
  - $\Sigma = \{1,2\}, \Gamma = \{0,1,2\}, Q = \{0,1,2\}$
- $\boldsymbol{\sigma = 0012}$

(q,x)	$\delta(q,x)$
$(0,0)$	$(0,0,\rightarrow)$
$(0,1)$	$(2,1,-)$
$(0,2)$	$(1,2,-)$

Nastro 1: $\boldsymbol{(0\#,0\#)(0\#,0\#,\rightarrow);(0\#,1\#)(10\#,1\#,-);(0\#,10\#)(1\#,10\#,-);\sqcup\sqcup\cdots}$

Nastro 2: $\boldsymbol{0\#0\#1\#10\#\ \sqcup\sqcup\cdots}$

Nastro 3: $\boldsymbol{10\#\ \sqcup\cdots}$

Lo stato ragiunto è Terminale! Accetta la Stringa Corrente

Macchina di Turing Universale

- Grazie alla costruzione definita in precedenza, possiamo dimostrare

- **Teorema.** Esiste una 3-TM U tale che, per ogni Macchina di Turing M e input σ per M la macchina U accetta $e(M,\sigma)$ se e solo se $\sigma \in L(M)$

- **Corollario 1**. Esiste una Macchina di Turing U' tale che, per ogni Macchina di Turing multi-nastro M e input σ per M la macchina U accetta $e(M,\sigma)$ se e solo se $\sigma \in L(M)$
  - Applicando la prova di equivalenza fra Macchine di Turing Multi-nastro e Macchine di Turing

- **Corollario 2**. Esiste una Macchina di Turing U' tale che, per ogni Macchina di Turing non-deterministica M e input σ per M la macchina U accetta $e(M,\sigma)$ se e solo se $\sigma \in L(M)$
  - Applicando la prova di equivalenza fra Macchine di Turing Non-Deterministiche e Macchine di Turing

Linguaggi e Problemi NON Turing Decidibili

Decidibilità dei Problemi

- **Problema Funzionale.** Sia fissata una funzione $f : \Sigma^* \rightarrow Y$ con Σ un alfabeto di simboli. Il **Problema Funzionale** P_f di f chiede, per ogni $a \in X$, il valore $f(a)$.
  - **Problema Decisionale.** Problema funzionale associato a una funzione booleana

- **Definizione**. Dato un problema decisionale P_f con definiamo il linguaggio $\mathcal{L}_f$ come segue $\mathcal{L}_f = \{x \in X \mid f(x) = 1\}$

- **Definizione**. Dato un problema decisionale P_f, diciamo che P_f è (Turin) Decidibile se $\mathcal{L}_f$ lo è, altrimenti P_f è detto (Turing) Indecidibile

Accept Problem

- **Definizione**. Il **Problema dell'Accettazione** (**Accept Problem**) è il problema funzionale associato alla funzione $\mathrm{ACCEPT}: \mathbb{M} \ \times \rightarrow \{0,1\}$ dove
  - $\mathbb{M}$ è l'insieme di tutti gli Encoding definiti in precedenza
  - $\mathrm{ACCEPT}\big(e(M,x)\big) = 1$ se e solo se M accetta l'input x

- A cui, ovviamente, corrisponde il seguente linguaggio

$$\boldsymbol{A = \{e(M,x) \mid M \ accetta \ l' \ input \ x\}}$$

Accept Problem è Semi-Decidibile

- **Proposizione**. Esiste una macchina di Turing M che **riconosce** A cioè $\mathrm{L(M)} = H$

- **Dimostrazione**. Dobbiamo presentare una macchina di Turing M' tale che, dato un qualunque $e(M,x) = \mathrm{d_M}, \mathrm{d_x},\ M'$ accetta $e(M,x)$ se e solo
  1. d_M è la descrizione di una macchina di Turing M,
  2. d_x è la descrizione di un input x per M e
  3. M accetta x

- Questa **macchina di Turing M' è essenzialmente la macchina di Turing universale U**.
  1. Verifica che l'input rappresenti un encoding corretto di una Macchina di Turing
  2. Lancia tale encoding utilizzando U e accetta se e solo sese la simulazione dell'encoding accetta

Accept Problem è Indecidibile – 1/3

- **Teorema**. Il linguaggio A non è decidibile
  - Non esiste una Macchina di Turing terminante M che riconosce A

- **Dimostrazione**. La dimostrazione è per assurdo, ovvero, supponiamo (per assurdo) che l'enunciato del teorema sia **falso** e ricaviamo una contraddizione alle ipotesi.

- Supponiamo che esista una Macchina di Turing $\boldsymbol{M_A}$ **che decide** $\boldsymbol{A}$
  - M_A è terminante (accetta o rifiuta ogni input) e
  - M_A accetta σ se e solo se $\sigma = [e(M); e(x)]$ e $\mathrm{x} \in L(M)$ **ovvero**
    1. $e(M)$ è la descrizione di una macchina di Turing M
    2. $e(x)$ è la descrizione di un input x per M
    3. $\boldsymbol{M}$ **accetta l'input** $\boldsymbol{x}$

Accept Problem è Indecidibile – 2/3

- **Teorema**. Il linguaggio A non è decidibile
  - Non esiste una Macchina di Turing terminante M che riconosce A

- **Dimostrazione**. Essendo M_A terminante, possiamo costruire una macchina F che, con input σ, simula M_A (utilizzando una Macchina di Turing universale) come segue
  - **Se M_A accetta l'input $[\sigma;\sigma]$** allora F **rifiuta l'input**
  - **Se M_A rifiuta l'input $[\sigma;\sigma]$** allora F **accetta l'input**

- Tale macchina può essere costruita da M_A semplicemente invertendo il comportamento della funzione di transizione verso gli stati q_{no} (che diventa q_{yes}) e q_{yes} (che diventa q_{no})

- **Nota**. F è una macchina terminante inquanto M_A lo è per assunzione

Accept Problem è Indecidibile – 3/3

- **Teorema**. Il linguaggio A non è decidibile
  - Non esiste una Macchina di Turing terminante M che riconosce A

- **Dimostrazione**. Per concludere la dimostrazione supponiamo di lanciare F con input $e(F)$
  - Lanciamo F sulla descrizione di F stessa
  - Questa configurazione è perfettamente lecita inquanto, essendo F una MT,
  - Possiamo quindi codificarla come una stringa applicando l'encoding definito in precedenza

- Dobbiamo considerare due casi distinti: F accetta o no.
  - **Supponiamo che F accetti l'input $e(F)$**. Quindi, F accetta $e(F)$ e, per definizione, M_A rifiuta l'input $[e(F); e(F)]$. Applicando la definizione di M_A, concludiamo che F non accetta l'input input $e(F)$, **contraddicendo** l'assunzione che F accetti $e(F)$
  - **Supponiamo che F non accetti l'input $e(F)$**. Quindi, per definizione, M_A accetta l'input $[e(F); e(F)]$. Applicando la definizione di M_A, concludiamo che F accetta con input $e(F)$, **contraddicendo** l'assunzione che F non accetti $e(F)$

Halting Problem

- **Definizione**. Il **Problema della Terminazione** (**Halting Problem**) è il problema funzionale associato alla funzione $\mathrm{HALTING}: \mathbb{M} \ \times \rightarrow \{0,1\}$ dove
  - $\mathbb{M}$ è l'insieme di tutti gli Encoding definiti in precedenza
  - $\mathrm{HALTING}\big(e(M,x)\big) = 1$ se e solo se M termina con input x

- A cui, ovviamente, corrisponde il seguente linguaggio

$$\boldsymbol{H = \{e(M,x) \mid M \ termina \ con \ input \ x\}}$$

L’Halting Problem è Semidecidibile

- **Proposizione**. Esiste una macchina di Turing M che riconosce H cioè $\mathrm{L(M)} = H$

- **Dimostrazione**. Dobbiamo presentare una macchina di Turing M' tale che, dato un qualunque $e(M, x) = \mathrm{d_M}, \mathrm{d_x}$, M' accetta $e(M, x)$ se e solo
  1. d_M è la descrizione di una macchina di Turing M,
  2. d_x è la descrizione di un input x per M e
  3. M termina con input x

- Questa **macchina di Turing M' è essenzialmente la macchina di Turing universale U**. Nello specifico, M' si ottiene da U con i seguenti passi
  1. Verifica che l’input rappresenti un encoding corretto di una Macchina di Turing
  2. Lancia tale encoding utilizzando U e accetta se la simulazione dell’encoding termina

L'Halting Problem è Indecidibile – 1/3

- **Teorema**. Il linguaggio H non è decidibile
  - Non esiste una Macchina di Turing terminante M che riconosce H

- **Dimostrazione**. La dimostrazione è (**di nuovo**) per assurdo, ovvero, supponiamo (per assurdo) che l'enunciato del teorema sia false e ricaviamo una contraddizione.

- Supponiamo che esista Macchina di Turing $\boldsymbol{M_H}$ **che decide** $\boldsymbol{H}$
  - M_H è terminante (accetta o rifiuta ogni input) e
  - M_H accetta σ se e solo se $\sigma = [e(M)e(x)]$ e M termina con input x **ovvero**
    - $e(M)$ è la descrizione di una macchina di Turing M
    - $e(x)$ è la descrizione di un input x per M
    - M termina con input x

L'Halting Problem è Indecidibile - 2/3

- **Teorema**. Il linguaggio H non è decidibile
  - Non esiste una Macchina di Turing terminante M che riconosce H

- **Dimostrazione**. Essendo M_H terminante, possiamo costruire una macchina F che, con input σ, simula M_H (utilizzando una Macchina di Turing universale) come segue
  - **Se M_H accetta $[\sigma;\sigma]$** allora F entra nello stato speciale q_{loop} in cui la testina continua a muoversi verso destra all'infinito senza terminare (M_H non termina con input σ)
  - **Se M_H rifiuta $[\sigma,\sigma]$** allora F entra nello stato q_{no} **(rifiuta l'input)**

- Tale macchina può essere costruita da M_H semplicemente cambiando il comportamento della funzione di transizione verso lo stato q_{yes} (che diventa q_{loop})

L’Halting Problem è Indecidibile - 3/3

- **Teorema**. Il linguaggio H non è decidibile
  - Non esiste una Macchina di Turing terminante M che riconosce H

- **Dimostrazione**. Per concludere la prova supponiamo di lanciare F con input $e(F)$
  - Lanciamo F sulla descrizione di F stessa
  - Questa configurazione è perfettamente lecita inquanto, essendo F una MT,
  - possiamo codificarla come una stringa applicando l’ecoding che abbiamo definito in precedenza

- Dobbiamo considerare due casi distinti: F termina o no.
  - **Supponiamo che F termini**. Quindi, F rifiuta $e(F)$ e, per definizione, M_H rifiuta l’input $[e(F); e(F)]$. Applicando la definizione di M_H, concludiamo che F non termina con input $e(F)$, **contraddicendo** l’assunzione che F termini
  - **Supponiamo che F non termini**. Quindi, per definizione, M_H accetta l’input $[e(F); e(F)]$. Applicando la definizione di M_H, concludiamo che F termina con input $e(F)$, **contraddicendo** l’assunzione che F non termini

Riduzioni di Karp

Indecidibilità per Riduzione – Intuizione

- Intuitivamente, se un linguaggio $\mathcal{L}$ non è decidibile, anche linguaggi simili non lo saranno
  - Tutti quei linguaggi che potrebbero aiutarci a decidere $\mathcal{L}$

- Esempio: $\boldsymbol{AT = \{e(M) \mid M(x)}$ **termina per ogni input** $\boldsymbol{x}$ **per** $\boldsymbol{M\}}$
  - Se AT fosse decidibile, anche H lo sarebbe
  - Una macchina per AT può essere utilizzata per decidere H (con qualche modifica…)

- Per formalizzare questa intuizione, introduciamo la nozione di riduzione
  - Funzioni calcolabili che convertono un linguaggio in un altro

La Nozione di Riduzione di Karp – Definizione

- **Definizione.** Una funzione $f\colon\Sigma^* \rightarrow \Sigma^*$ (con Σ un alfabeto di simboli) è ***computabile*** se esiste una **MdT** M_f tale che, per ogni $x \in \Sigma^*$, esiste una sequenza di configurazioni $C_1, \dots, C_n$ di M_f con $C_1 = (\epsilon, q_0, x)$, $C_n = \left(\epsilon, q_{yes}, f(x)\right)$ e $C_i \Rightarrow_M C_{i+1}, i = 1, \dots, n-1$

- **Definizione.** Diciamo che la macchina $\boldsymbol{M_f}$ **calcola** $\boldsymbol{f}$
  - **Intuizione**. Per ogni $x \in \Sigma^*$, $\boldsymbol{M_f}$ termina in una configurazione in cui il suo nastro contiene solo $f(x)$, effettivamente calcolando $f\colon\Sigma^* \rightarrow \Sigma^*$

- **Definizione**. Siano A e B due linguaggi sugli alfabeti Σ_{A} e Σ_B (non necessariamente gli distinti). ***A* è *Karp-riducibile* a *B*** ($\boldsymbol{A \leq_m B}$), se esiste una **funzione computabile** $f\colon\Sigma_{\mathrm{A}}^* \rightarrow \Sigma_{\mathrm{B}}^*$ tale che, per ogni $x \in \Sigma_A^*$, $x \in A$ se e solo se $f(x) \in B$
  - *Anche detto riducibile molti-a-uno*, oppure semplicemente ***riducibile***

- **Definizione**. La funzione f viene chiamata **Karp-Riduzione da $\boldsymbol{A}$ in $\boldsymbol{B}$**
  - Anche riduzione molti a uno oppure semplicemente ***riduzione***

La Nozione di Riduzione di Karp – Intuizione

- **Intuizione**. Se ($\boldsymbol{A \leq_m B}$) allora esiste un algoritmo P che, con input $x \in \Sigma_A$,
  - se $x \in A$, allora P restituisce una stringa in B
  - se $x \notin A$, allora P restituisce una stringa in Σ_B ma non in B

- La nozione di riduzione ci fornisce uno strumento per dimostrare decidibilità e indecidibilità di un linguaggio a partire dalla decidibilità o indecidibilità di un altro

1. Se B è **decidibile** e $\boldsymbol{A \leq_m B}$ allora A è **decidibile**
   - Per composizione, lanciamo prima la riduzione poi la macchina che decide B
2. Se A è **indecidibile** e $\boldsymbol{A \leq_m B}$ allora B è **indecidibile**
   1. Per composizione, se B fosse decidibile potremmo utilizzare la macchina che decide B per decidere anche A dopo aver calcolato la riduzione!

Indecidibilità per Riduzione

- **Lemma [Indecidibilità per Riduzione]**. Siano A e B due linguaggi sugli alfabeti Σ_A e Σ_B, rispettivamente, tale per cui $A \leq_m B$. Se A non è Turing Decidibile anche B non è Turing Decidibile

- **Dimostrazione**. Essendo $A \leq_m B$, esista una funzione $f\colon \Sigma_\mathrm{A} \to \Sigma_\mathrm{B}$ tale che $f(x) \in B$ se e solo $x \in A$, per ogni $x \in \Sigma_A^*$ e una macchina di Turing M_f che calcola f
- Supponiamo ora per assurdo che B sia decidibile
  - Esiste quindi una macchina M_B che decide B (M_B è terminante)
- Possiamo quindi costruire una Macchina di Turing M tale che
  1. M utilizza la macchina che calcola f per ottenere $f(x)$ da x
  2. M verifica se $f(x) \in B$ utilizzando la macchina M_B
  3. Se la risposta del Passo 2 è positiva, accetta
  4. Altrimenti rifiuta
- È facile osservare che la macchina M decide A

AlwaysTerminate

Problema:	AlwaysTerminate
INPUT:	La descrizione di una macchina di Turing M
OUTPUT:	1, se M termina su ogni input σ, 0 altrimenti

A cui, ovviamente, corrisponde il seguente linguaggio:

$$\boldsymbol{AT = \{e(M)\ \mid M(x)\ \textbf{termina per ogni input}\ x\ \textbf{per}\ M\}}$$

- **Teorema**. Il linguaggio AT non è Turing Decidibile
- **Dimostrazione.** La prova consiste in due passi

1. Forniamo una riduzione da H ad AT, dimostriamo, cioè $H \leq_m AT$
2. Applichiamo il Lemma dell'Indecidibilità per Riduzione

AlwaysTerminate

- **Lemma**. $H \leq_M AT$
- **Dimostrazione.** Dimostriamo l'esistenza di una riduzione f da H ad AT
- Sia $\tau = e(M); e(\sigma)$ un generico input per $HALTING$ dove Σ è l'alfabeto di input della macchina M
- $f(\tau)$ è la descrizione di una MT M' con alfabeto di input Σ che opera come segue
  - Con input una stringa $x \neq \sigma$, la machina M' termina rifiutando
  - Con input σ, la macchina M' simula M

- Dobbiamo dimostrare due proprietà di f
  1. f è computabile
  2. $f(x) \in AT$ se e solo se $x \in H$

- **Proprietà 1 (computabilità).** $f(\tau)$ può essere ottenuta semplicemente modificando l'encoding $e(M)$. Semplicemente, aggiungiamo alla descrizione di M una parte iniziale tale che
  1. Verifica se l'input x è uguale a σ
  2. Se lo è, lancia M, altrimenti rifiuta

AlwaysTerminate

- **Lemma**. $H \leq_M AT$
- **Dimostrazione.** Dimostriamo l'esistenza di una riduzione f da H ad AT
- Sia $\tau = e(M); e(\sigma)$ un generico input per $HALTING$ dove Σ è l'alfabeto di input della macchina M
- $f(\tau)$ è la descrizione di una MT M' con alfabeto di input Σ che opera come segue
  - Con input una stringa $x \neq \sigma$, la machina M' termina rifiutando
  - Con input σ, la macchina M' simula M

- **Proprietà 2 (riduzione).** Esaminiamo i due casi separatamente
- $\boldsymbol{\tau \in H}$, ovvero M termina con input σ. In questo caso, M' termina con input σ (perché M' simula M su σ) e termina per ogni altro input per definizione. Concludiamo $f(\tau) \in AT$
- $\boldsymbol{\tau \notin H}$, ovvero M non termina con input σ. In questo caso, M' non termina con input σ (perché M' simula M su σ) e quindi non termina per almeno un input. Concludiamo $f(\tau) \notin AT$

- Date **Proprietà 1** e **Proprietà 2**, possiamo conclude che f è una riduzione da H ad AT
- Questo dimostra $H \leq_M AT$

Teorema di Rice

Problema dell'Appartenenza

- **Definizione.** Sia Σ un alfabeto. Una **Proprietà dei Linguaggi di Σ** è una famiglia $\mathcal{P} \subseteq \mathrm{P}(\Sigma^*)$
  – Un sotto-insieme di tutti i linguaggi definibili sull'alfabeto Σ
  – Tutti quelli che posseggono la proprietà desiderata

- **Definizione.** Una **Proprietà dei Linguaggi di Σ** è **Triviale** se $\mathcal{P} = \emptyset$ o $\mathcal{P} = P(\Sigma^*)$
  – $\mathcal{P}$ è triviale se contiene tutti o nessun linguaggio di Σ

- **Definizione**. Il **Problema dell'Appartenenza per una Proprietà $\mathcal{P}$ dei Linguaggi di Σ** (**Belongs $\mathcal{P}$ Problem**) è il problema associato alla funzione $\mathrm{BELONGS}[\mathcal{P}]: \mathbb{M} \ \times \to \{0,1\}$ dove
  – $\mathbb{M}$ è l'insieme di tutti gli Encoding delle macchine di Turing definiti in precedenza
  – $\mathrm{BELONGS}[\mathcal{P}](e(M)) = 1$ se e solo se M riconosce uno dei linguaggi in $\mathcal{P}$

- A cui, ovviamente, corrisponde il seguente linguaggio

$$B_{\mathcal{P}} = \{e(M) \in \mathbb{M} \mid M\ riconosce\ un\ linguaggio\ in\ \mathcal{P}\}$$

Problema dell'Appartenenza – Esempio

- **Intuizione.** $\mathrm{BELONGS}[\mathcal{P}]$ chiede se un pezzo di codice riconosce un linguaggio che possiede la proprietà $\mathcal{P}$
  - Dobbiamo risolverlo ogni volta che vogliamo costruire un programma che analizza del codice
  - Ad esempio, se chiediamo ad un LLM di analizzare del codice scritto da noi, gli stiamo chiedendo di risolvere il problema dell'appartenenza (è vero che il mio codice risolvere il problema X?)

- **Esempio 1.** Vogliamo costruire un programma che analizza il codice di una funzione I in input e verifica se I ritorna $true$ se e solo se il suo input è un grafo connesso
  - Famiglia $\mathcal{C} = \{\, C \mid C \text{ è } un\ linguaggio\ di\ stringhe\ che\ rappresentano\ grafi\ connessi\}$
  - $\mathrm{BELONGS}[\mathcal{C}]$ risolve il problema desiderato

- **Esempio 2.** Vogliamo costruire un programma che analizza il codice di una funzione I in input e verifica se I ritorna $true$ se e solo se il suo input è una formula proposizionale su un certo alfabeto di variabili V
  - Famiglia $\mathcal{F} = \{\, F \mid F \text{ è } un\ linguaggio\ di\ stringhe\ che\ rappresentano\ formulae\ proposizionali\ su\ V\}$
  - $\mathrm{BELONGS}[\mathcal{F}]$ risolve il problema desiderato

Teorema di Rice (1/4)

- **Teorema[Rice]**. Sia $\mathcal{P}$ una Proprietà dei Linguaggi di Σ. Se $\mathcal{P}$ non è triviale allora $B_{\mathcal{P}}$ non è Turing Decidibile.

- **Dimostrazione.** L'idea della dimostrazione è che, per ogni $\mathcal{P}$ non triviale, esiste una riduzione $A \leq_M B_{\mathcal{P}}$ dove A è il problema dell'accettazione

Indecidibilità per il Teorema di Rice – Esempio

- **Definizione**. Sia $INFINITE : \mathbb{M} \rightarrow \{0,1\}$ la funzione tale che $INIFINITE(m) = 1$ se m è l'Encoding di una Macchina di Turing che accetta **un linguaggio di cardinalità infinita** su un alfabeto Σ

- **Definizione**. $INFINITE$ non è Turing Decidibile
- **Dimostrazione**. Sia $\mathcal{P}$ la famiglia dei linguaggi su Σ di cardinalità infinita. Osserviamo quanto segue:
  - $\mathcal{P}$ non è vuoto (esiste sempre un linguaggio di cardinalità infinita)
  - $\mathcal{P}$ non coincide con $P(\Sigma^*)$ (esiste sempre un linguaggio di cardinalità finita)
  - Concludiamo che **$\mathcal{P}$ non è una proprietà triviale dei linguaggi di Σ**
- Applicando il Teorema di Rice, concludiamo che il seguente linguaggio non è Turing Decidibile

$$B_{\mathcal{P}} = \{\, m \mid m \text{ è } l'encoding\ di\ una\ MT\ che\ riconosce\ un\ linguaggio\ di\ cardinalit\grave{a}\ infinita \,\}$$

- Ne consegue che il problema decisionale associato a $INIFINITE$ è indecidibile

Indecidibilità per il Teorema di Rice – Esempio

- **Definizione**. Sia $FINITE : \mathbb{M} \rightarrow \{0,1\}$ la funzione tale che $FINITE(m) = 1$ se m è l'Encoding di una Macchina di Turing che accetta **un linguaggio di cardinalità finita** su un alfabeto Σ

- **Definizione**. $FINITE$ non è Turing Decidibile
- **Dimostrazione**. Sia $\mathcal{P}$ la famiglia dei linguaggi su Σ di cardinalità finita. Osserviamo quanto segue:
  - $\mathcal{P}$ non è vuoto (esiste sempre un linguaggio di cardinalità finita)
  - $\mathcal{P}$ non coincide con $P(\Sigma^*)$ (esiste sempre un linguaggio di cardinalità infinita)
  - Concludiamo che **$\mathcal{P}$ non è una proprietà triviale dei linguaggi di Σ**
- Applicando il Teorema di Rice, concludiamo che il seguente linguaggio non è Turing Decidibile

$$B_{\mathcal{P}} = \{\, m \mid m \text{ è } l'encoding\ di\ una\ MT\ che\ riconosce\ un\ linguaggio\ di\ cardinalit\grave{a}\ finita \,\}$$

- Ne consegue che il problema decisionale associato a $FINITE$ è indecidibile

Indecidibilità per il Teorema di Rice – Esempio

- **Definizione**. Sia $REGULAR : \mathbb{M} \rightarrow \{0,1\}$ la funzione tale che $REGULAR(m) = 1$ se m è l'Encoding di una Macchina di Turing che accetta un **linguaggio regolare** su un alfabeto $\Sigma = \{\mathrm{a}, \mathrm{b}\}$

- **Definizione**. $REGULAR$ non è Turing Decidibile
- **Dimostrazione**. Sia $\mathcal{P}$ la famiglia dei linguaggi regolari su Σ. Osserviamo quanto segue:
  - $\mathcal{P}$ non è vuoto (esiste sempre un linguaggio regolare su Σ, possiamo definirlo con un ASFD)
  - $\mathcal{P}$ non coincide con $P(\Sigma^*)$ (esiste sempre un linguaggio non regolare su Σ, possiamo dimostrarlo col Pumping Lemma per i linguaggi regolari)
  - Concludiamo che **$\mathcal{P}$ non è una proprietà triviale dei linguaggi di Σ**
- Applicando il Teorema di Rice, concludiamo che il seguente linguaggio non è Turing Decidibile

$$B_{\mathcal{P}} = \{\, m \mid m \text{ è } l'encoding\ di\ una\ MT\ che\ riconosce\ un\ linguaggio\ regolare\}$$

- Ne consegue che il problema decisionale associato a $REGULAR$ è indecidibile

Indecidibilità per il Teorema di Rice – Esempio

- **Definizione**. Sia $CF : \mathbb{M} \rightarrow \{0,1\}$ la funzione tale che $CF(m) = 1$ se m è l'Encoding di una Macchina di Turing che accetta un **linguaggio non contestuale** su un alfabeto $\Sigma = \{\mathrm{a}, \mathrm{b}\}$

- **Definizione**. CF non è Turing Decidibile
- **Dimostrazione**. Sia $\mathcal{P}$ la famiglia dei linguaggi regolari su Σ. Osserviamo quanto segue:
  - $\mathcal{P}$ non è vuoto (esiste sempre un linguaggio regolare su Σ, possiamo definirlo con un Automa a Pila)
  - $\mathcal{P}$ non coincide con $P(\Sigma^*)$ (esiste sempre un linguaggio non regolare su Σ, possiamo dimostrarlo col Pumping Lemma per i linguaggi non contestuali)
  - Concludiamo che **$\mathcal{P}$ non è una proprietà triviale dei linguaggi di Σ**
- Applicando il Teorema di Rice, concludiamo che il seguente linguaggio non è Turing Decidibile

$$B_{\mathcal{P}} = \{\, m \mid m \text{ è } l'encoding\ di\ una\ MT\ che\ riconosce\ un\ linguaggio\ non\ contestuale\}$$

- Ne consegue che il problema decisionale associato a CF è indecidibile

Altri Problemi Indecidibili
che
Non Coinvolgono le Macchine

Post Correspondence Problem (PCP)

- **Post Correspondence Problem.** Il problema della corrispondenza di Emil Post (il matematico americano che lo ha scoperto)

- **Definizione**. Sia PCP: $\Sigma^* \rightarrow \{0,1\}$ la funzione tale che date due sequenze $x_1, \dots, x_k$ e $y_1, \dots, y_k$ di k stringhe su un alfabeto Σ , ritorna 1 se esiste una sequenza $i_1, \dots, i_m$ di $m \geq 1$ numeri interi compresi tra 1 e k tale per cui $x_{i_1} \dots x_{i_m} = y_{i_1} \dots y_{i_m}$

- **Esempio**: $x_1 = a, \quad x_2 = ab, \quad x_3 = bba, \qquad y_1 = baa, \quad y_2 = aa, \quad y_3 = bb$
- In questo caso $x_1; x_2; x_3; y_1; y_2; y_3 \in PCP$ perché la sequenza $(3, 2, 3, 1)$ è tale che
$$x_3x_2x_3x_1 = y_3y_2y_3y_1 = bbaabbbaa$$
- **Esempio**: $x_1 = bb, \quad x_2 = ab, \quad x_3 = c, \qquad y_1 = b, \quad y_2 = ba, \quad y_3 = bc$
- In questo caso $x_1; x_2; x_3; y_1; y_2; y_3 \in PCP$ perché la sequenza $(1, 2, 2, 2, 3)$ è tale che
$$x_1x_2x_2x_2x_3 = y_1y_2y_2y_2y_3 = bbababab c$$

- **Proposizione**. PCP non è Turing Decidibile
- Dimostrazione. La dimostrazione (non banale) prova una riduzione da $ACCEPT$

Decimo Problema di Hilbert

- **Decimo Problema di Hilbert.** Nell’anno 1900 Hilbert pubblicò una lista di 23 problemi rilevanti per la matematica contemporanea tra cui quello di trovare una procedura automatizzata per risolvere le equazioni diofantee (equazioni a coefficienti e soluzioni intere)
- **Definizione**. Una equazione polinomiale è diofantea se ha coefficienti interi e ammette almeno una soluzione intera (in $\mathbb{N}^k$).
- **Definizione**. Sia $\mathrm{H10}: \Sigma^* \to \{0,1\}$ tale che $H10(x) = 1$ se e solo se x rappresenta una equazione diofantea
  – Il problema si "riduce" a verificare che esista almeno una soluzione intera
  – Ovviamente dobbiamo fissare un Encoding
- **Proposizione**. $\mathrm{H10}$ non è Turing Decidibile
- **Dimostrazione**. La dimostrazione (molto complessa) prova una riduzione da $ACCEPT$
  – Ma deve farlo codificando macchine solo utilizzando polinomi … ☺
- **Nota** invece se ammettiamo soluzioni frazionarie in $\mathbb{Q}^k$, il problema diventa **decidibile** coi metodi del simplesso e dell’ellissoide.

Equivalenza fra Automi

- Fissiamo un Encoding da automi a pila a stringhe e
  - Possiamo utilizzare lo stesso che abbiamo utilizzato per definire le Macchine di Turing universali

- **Definizione**. Sia $EQ_{PILA}: \Sigma^* \rightarrow \{0,1\}$ tale che $EQ\ (x) = 1$ se e solo se $x = e(A); e(B)$ rappresenta una coppia di automi a pila A e B tali che $L(A) = L(B)$

- **Proposizione**. EQ_{PILA} non è Turing Decidibile ☹

- **Definizione**. Sia $EQ_{ASF}: \Sigma^* \rightarrow \{0,1\}$ tale che $EQ\ (x) = 1$ se e solo se $x = e(A); e(B)$ rappresenta una coppia di Automi a Stati Finiti A e B tali che $L(A) = L(B)$

- **Proposizione**. EQ_{ASF} è Turing Decidibile ☺

Linguaggi non Turing Riconoscibili

Linguaggi Turing Riconoscibili

- Fino ad ora abbiamo dimostrato che esistono linguaggi non Turing Decidibili
  - Linguaggi $\mathcal{L}$ per cui non esiste una Macchina di Turing **terminante** M tale che $\mathcal{L} = L(M)$

- **Domanda**. Esistono linguaggi non Turing riconoscibili?
  - Esiste un linguaggio $\mathcal{L}$ per cui non esiste una Macchina di Turing M tale che $\mathcal{L} = L(M)$
  - Anche se M non termina

Linguaggi NON Turing Riconoscibili – 1/2

- La risposta alla nostra domanda è positiva ovvero tali linguaggi esistono
  - È una conseguenza diretta della seguente semplice osservazione:

- **Teorema**: Sia $\mathcal{L}$ un linguaggio di stringhe sull'alfabeto Σ. Allora $\mathcal{L}$ è decidibile se e solo se sia $\mathcal{L}$ che $\bar{\mathcal{L}} = \{s \in \Sigma^* \mid s \notin \mathcal{L}\}$ sono semi-decidibili

- **Dimostrazione**. Dobbiamo provare due enunciati diversi.
- ***Se $\mathcal{L}$ è decidibile allora entrambi $\mathcal{L}$ e $\bar{\mathcal{L}}$ sono semi-decidibili.*** *Se $\mathcal{L}$ è decidibile, allora esiste una MT* M tale che M accetta s se e solo se $s \in \mathcal{L}$ e M termina su ogni input. Per decidere $\bar{\mathcal{L}}$, semplicemente definiamo la macchina $\bar{M}$ tale che $\bar{M}$ accetta s se e solo se M rifiuta s

Linguaggi NON Turing Riconoscibili – 2/2

- ***Se $\mathcal{L}$ e $\bar{\mathcal{L}}$ sono semi-decidibili, allora $\mathcal{L}$ è decidibile.*** Siano M ed $\bar{M}$ le macchine di Turing che accettano, rispettivamente, $\mathcal{L}$ e $\bar{\mathcal{L}}$ (non necessariamente terminanti)
  - Nota che sia M che $\bar{M}$ condividono lo stesso alfabeto di input Σ

- Considera la macchina di Turing a due nastri M' che, dato un qualsiasi input $x \in \Sigma^*$
  1. Copia x sul secondo nastro
  2. Simula il comportamento di M su x usando Nastro 1 e **accetta** se M accetta x
  3. Simula il comportamento di $\bar{M}$ su x usando Nastro 2 e **rifiuta** se $\bar{M}$ accetta x

- Ora osserviamo quanto segue
  - M accetta (terminando) tutte le $s \in \mathcal{L}$, quindi M' **accetta** (terminando) tute le $s \in \mathcal{L}$
  - $\bar{M}$ accetta (terminando) tutte le $s \notin \mathcal{L}$, quindi M' **rifiuta** (terminando) tute le $s \notin \mathcal{L}$

- Possiamo concludere che $\mathcal{L}$ è decidibile a causa di M'

Linguaggi NON Turing Riconoscibili – Esempio

- **Corollario**: Se $\mathcal{L}$ è Turing Riconoscibile ma non Turing Decidibile allora $\bar{\mathcal{L}}$ non è Turing Riconoscibile

- **Esempio 1**. Il complemento $\bar{A}$ del linguaggio A è definito come segue

$$\bar{A} = \{d_M; d_x \mid d_M \text{ non è la descrizione di una macchina di Turing } M \text{ OPPURE } d_M \text{ non accetta l'input } d_x\}$$

- **Proposizione 1**. Il linguaggio $\bar{A}$ non è Turing Riconoscibile

- **Esempio 2**. Il complemento $\bar{H}$ del linguaggio H è definito come segue

$$\bar{H} = \{d_M; d_x \mid d_M \text{ non è la descrizione di una macchina di Turing } M \text{ OPPURE } d_M \text{ non termina con input } d_x\}$$

- **Proposizione 2**. Il linguaggio $\bar{H}$ non è Turing Riconoscibile