Lezione di Informatica Teorica

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione e Recap

Questa lezione riprende e approfondisce i concetti introdotti precedentemente, con un focus sulla formalizzazione dei problemi e l'introduzione di un modello di calcolo più potente: la Macchina di Turing.

1.1 Problemi e Linguaggi

Abbiamo studiato la complessità non degli algoritmi, ma dei **problemi**. La differenza sostanziale è che qui formalizziamo i problemi stessi.

Esistono due tipi principali di problemi:

- **Problemi di Ricerca**: Le risposte possono essere varie.
- **Problemi di Decisione**: La risposta è sempre un booleano (Sì/No).

I problemi di decisione possono riguardare vari contesti (matrici, grafi, immagini, testo, ecc.). Per semplificare l'analisi, si scelgono problemi che possono essere formalizzati in modo uniforme. Abbiamo scelto di studiare i **problemi di decidere linguaggi**.

Definizione 1 (Decidere un linguaggio)**.** *Decidere un linguaggio significa verificare se una data stringa appartiene o meno a un determinato linguaggio.*

- ***Input**: Una stringa.*
- ***Linguaggio**: Fa parte della definizione del problema, non dell'input.*
- ***Output**: Sì/No (la stringa appartiene al linguaggio?).*

Codifica dei Problemi in Linguaggi: È possibile ricodificare problemi di decisione arbitrari in problemi di decisione di linguaggi.

Esempio 1 (Problema dei Grafi Totalmente Connessi)**.** ***Problema**: Dato un grafo, stabilire se è totalmente connesso.*

- *Questo è un problema di decisione (risposta Sì/No).*
- ***Codifica in un linguaggio**: Possiamo inventare un alfabeto Σ e una codifica tale per cui le stringhe che fanno parte del linguaggio L_{GC} sono solo quelle stringhe che, secondo la nostra codifica, rappresentano grafi totalmente connessi.*
- *Una stringa in input che non codifica un grafo (secondo la nostra codifica) non appartiene a L_{GC}.*
- *Le stringhe che codificano grafi non totalmente connessi non appartengono a L_{GC}.*

In questo modo, decidere se una stringa appartiene a L_{GC} equivale a decidere se il grafo codificato è totalmente connesso.

Questa codifica può essere estesa anche ai problemi di ricerca:

- Un problema di ricerca è una relazione binaria tra stringhe (input-output).
- Per codificare un problema di ricerca in un linguaggio, il linguaggio conterrebbe stringhe che codificano le coppie (input, output) valide per quel problema. Decidere l'appartenenza di una stringa a tale linguaggio implicherebbe, in qualche modo, il calcolo della soluzione.

Per semplicità, in questo corso ci concentreremo sui problemi di decisione e sulla loro formalizzazione come problemi di decisione di linguaggi.

1.2 Automi a Stati Finiti (Recap)

Per determinare se un linguaggio è decidibile (intuitivamente, se esiste un algoritmo che lo risolve), abbiamo introdotto i modelli di calcolo. Inizialmente, abbiamo visto gli **Automi a Stati Finiti (FA)**.

- **DFA (Deterministic Finite Automata)**: Per ogni stato e simbolo in input, esiste una sola transizione possibile.
- **NFA (Non-Deterministic Finite Automata)**: Per ogni stato e simbolo in input, possono esserci più transizioni possibili. Un NFA accetta una stringa se esiste *almeno un percorso* di computazione che porta a uno stato accettante consumando tutto l'input.

Potere Computazionale degli FA: Gli NFA e i DFA hanno lo stesso potere computazionale. Ogni NFA può essere convertito in un DFA equivalente. Il tempo di esecuzione è lineare rispetto alla lunghezza dell'input.

Esempio 2 (Linguaggio $L = \{a^n b^n \mid n \geq 0\}$)**.** *Questo linguaggio, che rappresenta stringhe con n 'a' seguite da n 'b' (es.* `aabb`*,* `aaabbb`*),* ***non può essere riconosciuto da un automa a stati finiti****.*

- ***Intuito****: Un FA non ha memoria sufficiente per "contare" le 'a' e confrontarle con le 'b'. Ha un numero finito di stati, quindi non può memorizzare un numero arbitrario n.*

Questo implica la necessità di un modello di calcolo più potente.

1.3 Linguaggi Regolari ed Espressioni Regolari

Gli automi a stati finiti (DFA/NFA) sono in grado di riconoscere tutti e soli i **linguaggi regolari**.

Definizione 2 (Linguaggio Regolare ed Espressioni Regolari)**.** *Sia Σ un alfabeto. Un linguaggio regolare su Σ è un linguaggio le cui stringhe possono essere descritte da* ***espressioni regolari****. Le regole per costruire espressioni regolari sono:*

- *Se $\alpha \in \Sigma$, allora α è un'espressione regolare (es.* 0, 1*).*
- *Se α e β sono espressioni regolari, allora anche la loro* ***concatenazione*** *$\alpha\beta$ è un'espressione regolare (es.* 00, 0111*).*
- *Se α e β sono espressioni regolari, allora anche la loro* ***disgiunzione*** *$\alpha \vee \beta$ (spesso scritta $\alpha + \beta$ o $\alpha|\beta$) è un'espressione regolare. Rappresenta le stringhe che sono generate da α oppure da β (es.* $0|11$ *accetta* 0 *o* 11*).*
- *Se α è un'espressione regolare, allora α^* (Kleene Star) è un'espressione regolare. Rappresenta una concatenazione di zero o più occorrenze di α (es.* $(10)*$ *accetta ϵ,* 10, 1010, *...).*
- *A volte si usa α^+ (Kleene Plus) per indicare una concatenazione di una o più occorrenze di α (es.* $(10)+$ *accetta* 10, 1010, *... ma non ϵ).*

Proprietà: I linguaggi riconoscibili dagli automi a stati finiti sono esattamente i linguaggi regolari. Il linguaggio $\{a^n b^n \mid n \geq 0\}$ **non è un linguaggio regolare**, motivo per cui gli FA non possono riconoscerlo.

2 Macchine di Turing (TM)

Per risolvere problemi più complessi come $\{a^n b^n \mid n \geq 0\}$, è necessario un modello di calcolo più potente: la Macchina di Turing.

2.1 Introduzione e Concetti Base

La Macchina di Turing è un modello astratto di calcolo ideato da Alan Turing negli anni '30 per formalizzare il concetto di "calcolabilità".

Differenze chiave rispetto agli Automi a Stati Finiti:

- **Nastro Infinito**: La TM opera su un nastro infinito in entrambe le direzioni, diviso in celle.
- **Simbolo Blank (`B`)**: Le celle vuote del nastro contengono un simbolo speciale (blank) per delimitare la stringa di input.
- **Testina di Lettura/Scrittura**: La TM ha una testina che può non solo leggere un simbolo, ma anche *scrivere* (sovrascrivere) un simbolo sulla cella corrente.
- **Movimento Bidirezionale della Testina**: La testina può spostarsi a destra (`R`) o a sinistra (`L`) lungo il nastro.

Queste capacità conferiscono alla TM una "memoria" infinita e la capacità di manipolare i dati sul nastro, superando le limitazioni degli FA.

Funzionamento: Una TM è un automa con un numero finito di stati. Ogni passo della computazione è determinato dallo stato corrente e dal simbolo letto dalla testina. La TM esegue le seguenti azioni:

1. Transisce a un nuovo stato.
2. Scrive un simbolo sulla cella corrente (sovrascrivendo quello precedente).
3. Sposta la testina a destra o a sinistra.

2.2 Definizione Formale della Macchina di Turing

Definizione 3 (Macchina di Turing). *Una Macchina di Turing (TM) M è una 7-tupla:* $M = \langle \Sigma, \Gamma, B, Q, q_0, F, \delta \rangle$ *dove:*

- Σ*: è l'alfabeto di input (insieme finito di simboli che possono apparire nella stringa di input).*
- Γ*: è l'alfabeto di nastro (insieme finito di simboli che possono essere scritti sul nastro). Deve essere* $\Sigma \subseteq \Gamma$.
- $B \in \Gamma \setminus \Sigma$*: è il simbolo di blank, non è parte dell'input e indica una cella vuota.*
- Q*: è l'insieme finito degli stati interni della macchina.*
- $q_0 \in Q$*: è lo stato iniziale.*
- $F \subseteq Q$*: è l'insieme degli stati accettanti (o finali).*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$*: è la funzione di transizione (parziale). Data una coppia (stato corrente, simbolo letto), restituisce una tripla (nuovo stato, simbolo da scrivere, direzione del movimento della testina).*

2.3 Esempio: Macchina di Turing per $L = \{a^n b^n \mid n \geq 0\}$

Strategia intuitiva: L'idea è "barrare" una 'a' dal lato sinistro e una 'b' dal lato destro ad ogni passo, tornando indietro e avanti fino a quando tutte le 'a' e 'b' sono state "consumate". Se il numero di 'a' e 'b' è lo stesso, il nastro sarà vuoto alla fine.

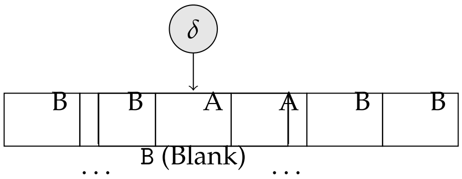

Figura 1: Rappresentazione del nastro della Macchina di Turing con testina.

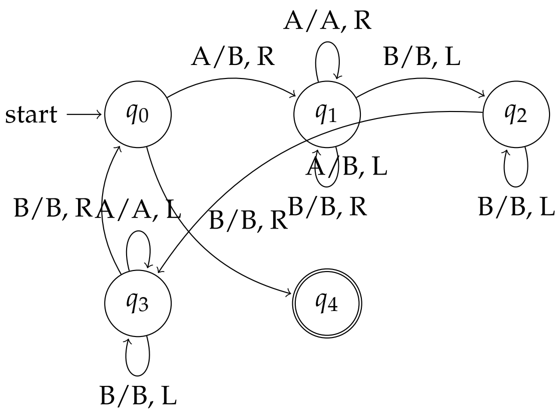

Figura 2: Diagramma di stati per la Macchina di Turing che riconosce $L = \{a^n b^n \mid n \geq 0\}$.
Nota: Le etichette degli archi sono nel formato: `SimboloLetto/SimboloScritto, DirezioneMovimento`.

Traccia del funzionamento (con correzioni al diagramma basate sulla discussione): 1. q_0 **(Stato Iniziale)**:

- Se legge `A`: Scrive `B` (blank), sposta a destra (`R`), va in q_1. (Cancella la prima 'a').
- Se legge `B` o `B` (blank, dopo aver cancellato tutto): Significa che non ci sono più 'a' da cancellare, quindi verifica se anche le 'b' sono finite. Se legge `B` e sposta a destra, va in q_4 (stato accettante).

2. q_1 **(Cerca l'ultima 'b')**:

- Se legge `A`: Scrive `A`, sposta a destra (`R`), rimane in q_1. (Salta le 'a' rimanenti).
- Se legge `B`: Scrive `B`, sposta a destra (`R`), rimane in q_1. (Salta le 'b' fino alla fine).
- Se legge `B` (blank): Scrive `B`, sposta a sinistra (`L`), va in q_2. (Ha raggiunto la fine della stringa, torna indietro per trovare l'ultima 'b').

3. q_2 **(Cancella l'ultima 'b')**:

- Se legge `B`: Scrive `B` (blank), sposta a sinistra (`L`), va in q_3. (Cancella l'ultima 'b' e inizia a tornare all'inizio della stringa).

4. q_3 **(Torna all'inizio)**:

- Se legge `A`: Scrive `A`, sposta a sinistra (`L`), rimane in q_3. (Salta le 'a' rimanenti, andando a sinistra).
- Se legge `B`: Scrive `B`, sposta a sinistra (`L`), rimane in q_3. (Salta le 'b' rimanenti, andando a sinistra).
- Se legge `B` (blank): Scrive `B`, sposta a destra (`R`), va in q_0. (Ha raggiunto l'inizio della stringa, torna a q_0 per la prossima iterazione).

5. q_4 **(Stato Accettante)**:

- Questo è uno stato accettante. Se la TM raggiunge q_4, la stringa è accettata.

Gestione di stringhe "strane" (e.g., `ABaAB`): La macchina come progettata dovrebbe rifiutare stringhe che non seguono il pattern a^*b^*.

- Se q_0 legge `B` all'inizio, si muove a q_4, quindi accetterebbe la stringa vuota ϵ. Ma se è B e non ϵ, si muoverebbe a q_4 e accetterebbe una stringa di soli Bs (e.g., B). Questo è un dettaglio che richiede un'attenzione specifica sul caso base ϵ o stringhe di soli Bs.
- Se in q_1 (dopo aver cancellato la prima 'a' e spostato a destra) la macchina incontra un'altra 'a' dopo una 'b', non ci sono transizioni definite per questa sequenza, e la macchina si blocca, rifiutando la stringa.

2.4 Computazione di una Macchina di Turing

Per formalizzare il comportamento di una TM, si introduce il concetto di **configurazione**.

Definizione 4 (Configurazione di una TM)**.** *Una **configurazione** di una TM M è una fotografia dello stato corrente di esecuzione della macchina. È rappresentata da una stringa che include:*

- *La parte non-blank del nastro.*
- *Lo stato corrente della TM.*
- *La posizione della testina sul nastro.*

La notazione comune è uqv, dove u è la stringa sul nastro a sinistra della testina, q è lo stato corrente, e v è la stringa sul nastro a destra della testina (incluso il simbolo letto dalla testina, che è il primo simbolo di v). Si omettono i blank a meno che non siano rilevanti per la posizione della testina.

Esempio 3.
- *Aq_1BB: La stringa sul nastro è `ABB`, la TM è nello stato q_1, e la testina sta leggendo il primo `B`.*
- *$ABBq_1$`B`: La stringa sul nastro è `ABB`, la TM è nello stato q_1, e la testina sta leggendo il primo blank a destra della stringa.*

Definizione 5 (Successore Legale di una Configurazione)**.** *Date due configurazioni C_1 e C_2 per una TM M, diciamo che C_2 è un **successore legale** (o raggiungibile in un passo) di C_1 rispetto a M, e scriviamo $C_1 \xrightarrow{M} C_2$, se C_2 è la configurazione che M raggiunge partendo da C_1 ed eseguendo un solo passo secondo la sua funzione di transizione δ.*

Esempio 4. *Per la TM di $a^n b^n$, se $C_1 = q_0 AABB$, allora $C_2 = Bq_1 ABB$ (dopo aver cancellato la prima A e mosso a destra).*

Definizione 6 (Configurazione Iniziale)**.** *La **configurazione iniziale** di una TM M su una stringa di input $w = w_1 w_2 \dots w_n$ è $q_0 w_1 w_2 \dots w_n$. Si assume che la testina sia sul primo simbolo di w.*

Definizione 7 (Configurazione Finale)**.** *Una **configurazione finale** è una configurazione C per la quale la funzione di transizione δ non è definita per la combinazione (stato di C, simbolo letto in C). In altre parole, la macchina si blocca.*

Definizione 8 (Configurazione Accettante)**.** *Una **configurazione accettante** è una configurazione finale il cui stato corrente appartiene all'insieme degli stati accettanti F.*

Definizione 9 (Configurazione Rifiutante)**.** *Una **configurazione rifiutante** è una configurazione finale il cui stato corrente **non** appartiene all'insieme degli stati accettanti F.*

Definizione 10 (Computazione Parziale)**.** *Una **computazione parziale** di una TM M è una sequenza di configurazioni $C_1, C_2, \dots, C_k$ tale che $C_i \xrightarrow{M} C_{i+1}$ per ogni $1 \leq i < k$.*

Definizione 11 (Computazione (Completa))**.** *Una **computazione** di una TM M su una stringa di input w è una computazione parziale $C_1, C_2, \dots, C_k$ tale che:*

- *C_1 è la configurazione iniziale di M su w.*
- *C_k è una configurazione finale.*

Definizione 12 (Computazione Accettante)**.** *Una **computazione accettante** di una TM M su una stringa w è una computazione $C_1, \dots, C_k$ dove C_k è una configurazione accettante.*

Definizione 13 (Linguaggio di una Macchina di Turing $L(M)$)**.** *Il **linguaggio di una Macchina di Turing** M, denotato $L(M)$, è l'insieme di tutte le stringhe w tali che la computazione di M su w è accettante. $L(M) = \{w \mid M \text{ accetta } w\}$ Se una TM non si ferma mai su una stringa w, allora w non appartiene a $L(M)$. Se si ferma in uno stato non accettante, w non appartiene a $L(M)$.*

2.5 Accettazione vs. Decisione di un Linguaggio

La differenza tra "accettare" e "decidere" un linguaggio da parte di una Macchina di Turing è fondamentale in Teoria della Computabilità.

Definizione 14 (Macchina di Turing che Decide un Linguaggio)**.** *Una Macchina di Turing M **decide** un linguaggio L se e solo se per ogni stringa $w \in \Sigma^*$:*

- *Se $w \in L$, allora M si arresta (termina) e accetta w (ovvero, la computazione termina in una configurazione accettante).*
- *Se $w \notin L$, allora M si arresta (termina) e rifiuta w (ovvero, la computazione termina in una configurazione rifiutante).*

*In questo caso, si dice che L è un **linguaggio decidibile**. Una macchina che decide garantisce una risposta (Sì o No) in tempo finito per ogni input.*

Definizione 15 (Macchina di Turing che Accetta un Linguaggio)**.** *Una Macchina di Turing M **accetta** un linguaggio L se e solo se per ogni stringa $w \in \Sigma^*$:*

- *Se $w \in L$, allora M si arresta e accetta w.*
- *Se $w \notin L$, allora M* ***non accetta*** *w. Ciò significa che M potrebbe arrestarsi e rifiutare w,* oppure potrebbe non arrestarsi mai (loop indefinitamente).

In questo caso, si dice che L è un ***linguaggio accettabile*** *(o ricorsivamente enumerabile).*

Implicazioni:

- La classe dei linguaggi decidibili è un sottoinsieme stretto della classe dei linguaggi accettabili.
- Se un linguaggio è decidibile, allora è anche accettabile.
- Esistono linguaggi che sono accettabili ma non decidibili. Per questi linguaggi, se la risposta è "Sì", la macchina terminerà e lo dirà. Ma se la risposta è "No", la macchina potrebbe entrare in un loop infinito e non fornire mai una risposta, lasciando l'utente in attesa indefinita. Questo è un problema fondamentale in Informatica Teorica.