Lezione di Informatica Teorica: Automi a Stati Finiti

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione e Recap

Questa lezione riprende i concetti introdotti precedentemente, approfondendo la definizione formale di automi a stati finiti (deterministi e non-deterministici) e le loro capacità computazionali.

1.1 Problemi di Ricerca e Problemi di Decisione

In informatica teorica, siamo principalmente interessati ai *problemi di ricerca*, che sono problemi generici in cui la risposta può essere qualsiasi valore valido (es. "Qual è il prodotto di $A \times B$?", "Qual è il percorso minimo tra due punti?"). Un sottoinsieme importante sono i *problemi di decisione*, in cui la risposta possibile è solo "sì" o "no". La relazione tra i due tipi è significativa: spesso, un problema di ricerca può essere trasformato in un problema di decisione correlato. Per l'analisi, ci focalizziamo sui problemi di decisione per la loro maggiore semplicità.

1.2 Linguaggi e Decidibilità

I problemi di decisione sono spesso legati alla *decidibilità dei linguaggi*.

Definizione 1.1 (Linguaggio)**.** *Un* linguaggio *è semplicemente un insieme di stringhe su un dato alfabeto* Σ*.*

Definizione 1.2 (Decidere un Linguaggio)**.** Decidere un linguaggio L *significa, data una stringa* w*, stabilire se* w *appartiene a* L *o meno. Le risposte possibili sono "sì" (se* $w \in L$*) o "no" (se* $w \notin L$*). Questo è un problema di decisione.*

Nota Importante: Quando si decide un linguaggio L, l'input al problema è la **stringa** w, non il linguaggio L stesso. Il linguaggio L fa parte della definizione del problema.

2 Automi a Stati Finiti Deterministici (DFA)

Per formalizzare la decisione dei linguaggi, utilizziamo il concetto di *automa*. Gli automi sono modelli di calcolo semplici e formalmente definibili.

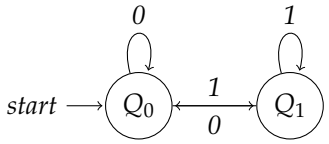
2.1 Intuizione e Funzionamento

Un automa a stati finiti può essere immaginato come un computer molto semplice che riceve l'input su un "nastro" (come uno scontrino). La macchina legge un simbolo per volta dal nastro, si sposta in un nuovo stato basandosi sul simbolo letto e sul suo stato corrente, e il simbolo letto viene "consumato" (non può essere riletto). Non si torna mai indietro sul nastro.

Esempio 2.1 (Riconoscimento di Numeri Dispari Binari)**.** *Consideriamo il linguaggio delle stringhe binarie che codificano numeri dispari. Un automa per questo linguaggio necessita solo di guardare l'ultimo bit. Possiamo usare due stati:* Q_0 *(l'ultimo simbolo visto era 0, o il numero parziale è pari) e* Q_1 *(l'ultimo simbolo visto era 1, o il numero parziale è dispari). Lo stato iniziale è* Q_0*.* Q_1 *sarà lo stato accettante.*

- *Se siamo in* Q_0 *e leggiamo* 0*, rimaniamo in* Q_0*.*
- *Se siamo in* Q_0 *e leggiamo* 1*, andiamo in* Q_1*.*
- *Se siamo in* Q_1 *e leggiamo* 0*, andiamo in* Q_0*.*

- *Se siamo in Q_1 e leggiamo* 1*, rimaniamo in Q_1.*



2.2 Definizione Formale di DFA

Definizione 2.1 (Automa a Stati Finiti Deterministico (DFA))**.** *Un automa a stati finiti deterministico D è una quintupla $(\Sigma, Q, q_0, F, \delta)$, dove:*

- *Σ è l'alfabeto di input (un insieme finito non vuoto di simboli).*
- *Q è un insieme finito di stati.*
- *$q_0 \in Q$ è lo stato iniziale.*
- *$F \subseteq Q$ è l'insieme degli stati finali (o accettanti).*
- *$\delta : Q \times \Sigma \to Q$ è la funzione di transizione. Per ogni coppia (stato corrente, simbolo letto), δ determina **un solo** prossimo stato.*

Il numero di stati in Q è fisso e non può cambiare durante la computazione.

2.3 Computazione di un DFA

Per capire il funzionamento di un automa, modelliamo i suoi passi attraverso il concetto di "configurazione".

Definizione 2.2 (Configurazione)**.** *Una* configurazione *di un DFA è una coppia (q, w), dove:*

- *$q \in Q$ è lo stato corrente in cui si trova l'automa.*
- *$w \in \Sigma^*$ è la porzione di stringa di input che deve ancora essere letta.*

Una configurazione rappresenta uno "snapshot" dello stato di avanzamento della computazione.

Definizione 2.3 (Computazione Parziale)**.** *Una* computazione parziale *di un DFA D su una stringa $w = c_1 c_2 \dots c_n$ è una sequenza di $m + 1$ configurazioni:*

$$(R_0, c_1 \dots c_n) \to (R_1, c_2 \dots c_n) \to \dots \to (R_m, c_{m+1} \dots c_n)$$

tale che:

- *$R_0 = q_0$ (la computazione inizia dallo stato iniziale con l'intera stringa in input).*
- *Per ogni $0 \leq i < m$, si ha $R_{i+1} = \delta(R_i, c_{i+1})$.*

Il simbolo c_{i+1} viene letto e consumato, e l'automa si sposta nello stato R_{i+1}.

Esempio 2.2 (Traccia di Computazione per 001)**.** *Usiamo il DFA per numeri dispari e la stringa $w =$* 001*.*

- *Passo 0:* $(Q_0, 001)$ *(Configurazione iniziale)*
- *Passo 1:* $(Q_0, 01)$ $(\delta(Q_0, 0) = Q_0)$
- *Passo 2:* $(Q_0, 1)$ $(\delta(Q_0, 0) = Q_0)$
- *Passo 3:* (Q_1, ϵ) $(\delta(Q_0, 1) = Q_1)$

L'ultima configurazione è (Q_1, ϵ)*.*

Definizione 2.4 (Computazione Completa)**.** *Una* computazione completa *(o semplicemente* computazione*) è una computazione parziale che è massimale, cioè non può essere estesa ulteriormente. Questo si verifica in due casi:*

- *La stringa di input è stata completamente consumata (la porzione residua è* ϵ*).*
- *Non esiste una transizione definita per lo stato corrente e il simbolo da leggere (l'automa si "blocca").*

Definizione 2.5 (Accettazione di una Stringa (DFA))**.** *Un DFA* D accetta *una stringa* w *se la sua unica computazione completa, che inizia dalla configurazione* (q_0, w)*, termina in una configurazione* (q_f, ϵ) *tale che* $q_f \in F$*. In altre parole, due condizioni devono essere soddisfatte:*

1. *L'intera stringa di input* w *deve essere consumata.*
2. *Lo stato finale raggiunto dall'automa deve essere uno stato accettante (*$q_f \in F$*).*

Definizione 2.6 (Rifiuto di una Stringa (DFA))**.** *Un DFA* D rifiuta *una stringa* w *se la computazione su* w *non è accettante. Ciò può avvenire se:*

- *L'automa termina in uno stato* $q \notin F$ *pur avendo consumato tutto l'input.*
- *L'automa si blocca (non ha transizioni definite) prima di aver consumato tutto l'input.*

Definizione 2.7 (DFA che Decide un Linguaggio)**.** *Un DFA* D decide *un linguaggio* L *se per ogni stringa* $w \in L$*,* D *accetta* w*, e per ogni stringa* $w \notin L$*,* D *rifiuta* w*.*

3 Automi a Stati Finiti Non-Deterministici (NFA)

Gli automi non-deterministici introducono il concetto di scelte multiple durante la computazione.

3.1 Introduzione al Non-Determinismo

Consideriamo il linguaggio delle stringhe binarie che terminano con 000, espresso dalla espressione regolare $(0|1)^*000$.

Esempio 3.1 (NFA per $(0|1)^*000$)**.** *Un tentativo intuitivo di DFA potrebbe essere:*

- Q_0*: Stato iniziale e per leggere* $(0|1)^*$*.*
- Q_1*: Dopo aver letto il primo* 0 *dei* 000 *finali.*
- Q_2*: Dopo aver letto il secondo* 0 *dei* 000 *finali.*
- Q_3*: Stato finale, dopo aver letto il terzo* 0 *dei* 000 *finali.*

Le transizioni sarebbero:

- *Da Q_0:*
  - *Se legge* 0*: può rimanere in Q_0 (per $(0|1)^*$)* oppure *andare in Q_1 (se è il primo* 0 *della sequenza finale). Questo è un punto di non-determinismo.*
  - *Se legge* 1*: rimane in Q_0.*
- *Da Q_1:*
  - *Se legge* 0*: va in Q_2.*
  - *Se legge* 1*: torna in Q_0 (la sequenza* 000 *è interrotta).*
- *Da Q_2:*
  - *Se legge* 0*: va in Q_3.*
  - *Se legge* 1*: torna in Q_0.*
- *Da Q_3:*
  - *Se legge* 0*: rimane in Q_3 (ma in realtà non è così semplice per $(0|1)^*$*000 *per via dei* 000 *finali). Per la semplicità dell'esempio del professore, si può immaginare che da Q_3 si accettino solo stringhe che finiscono esattamente con* 000 *e nessun altro carattere dopo. Se la stringa è* 010001 *dopo Q_3 dovrebbe andare in Q_0.*

Il problema evidenziato dal professore è il non-determinismo *in Q_0 quando legge* 0*: può andare in Q_0 o Q_1.*

Definizione 3.1 (Automa a Stati Finiti Non-Deterministico (NFA))**.** *Un automa a stati finiti non-deterministico N è una quintupla $(\Sigma, Q, q_0, F, \delta)$, dove:*

- *Σ, Q, q_0, F sono definiti come per i DFA.*
- *$\delta : Q \times \Sigma \to \mathcal{P}(Q)$ è la funzione di transizione (o relazione di transizione). Per ogni coppia (stato corrente, simbolo letto), δ determina un **insieme** di possibili prossimi stati ($\mathcal{P}(Q)$ denota l'insieme delle parti di Q).*

3.2 Computazione di un NFA

A causa delle transizioni che possono portare a più stati, per un dato input un NFA può avere *più computazioni differenti*.

Definizione 3.2 (Computazione Parziale per NFA)**.** *Una* computazione parziale *di un NFA N su una stringa $w = c_1 c_2 \dots c_n$ è una sequenza di $m + 1$ configurazioni:*

$$(R_0, c_1 \dots c_n) \to (R_1, c_2 \dots c_n) \to \dots \to (R_m, c_{m+1} \dots c_n)$$

tale che:

- *$R_0 = q_0$ (la computazione inizia dallo stato iniziale con l'intera stringa in input).*
- *Per ogni $0 \leq i < m$, si ha $R_{i+1} \in \delta(R_i, c_{i+1})$. (Il prossimo stato è uno qualsiasi tra quelli ammessi dalla relazione di transizione).*

Esempio 3.2 (Traccia di Computazione per 0100 su NFA $(0|1)^*000$)**.** *Consideriamo l'NFA precedente e la stringa* $w = 0100$*.*

- ***Percorso Accettante (se esistesse Q3 come accettante):***
- $(Q_0, 0100)$
- $\xrightarrow{\textit{leggi 0, transisci in } Q_0} (Q_0, 100)$
- $\xrightarrow{\textit{leggi 1, transisci in } Q_0} (Q_0, 00)$
- $\xrightarrow{\textit{leggi 0, transisci in } Q_1} (Q_1, 0)$
- $\xrightarrow{\textit{leggi 0, transisci in } Q_2} (Q_2, \epsilon)$
- $\xrightarrow{\textit{leggi } \epsilon\textit{, transisci in } Q_3} (Q_3, \epsilon)$ *Questo percorso, se* Q_3 *è accettante, porta all'accettazione.*
- ***Percorso Rifiutante:***
- $(Q_0, 0100)$
- $\xrightarrow{\textit{leggi 0, transisci in } Q_1} (Q_1, 100)$
- *Ora, da* Q_1 *leggendo* 1*, si torna in* Q_0*.*
- $\xrightarrow{\textit{leggi 1, transisci in } Q_0} (Q_0, 00)$ **A questo punto l'automa non è riuscito a finire la sequenza* 000 *nel modo sperato, e dovrebbe continuare a processare per vedere se trova altri* 000*.* *Esempio più semplice: se da* Q_1 *non ci fosse una transizione per '1', la macchina si bloccherebbe.**

Definizione 3.3 (Accettazione di una Stringa (NFA))**.** *Un NFA* N accetta *una stringa* w *se* ***esiste almeno una*** *computazione completa per* w *che sia accettante (ovvero, termina in uno stato finale* $q_f \in F$ *con input esaurito* ϵ*).*

Nota: Un NFA rifiuta una stringa w se *tutte* le possibili computazioni per w sono rifiutanti.

La Natura del Non-Determinismo

È cruciale comprendere che gli NFA sono *modelli astratti* di calcolo. Non implicano che una macchina fisica "provi tutte le strade contemporaneamente", "tiri a indovinare" o "faccia backtracking". La loro utilità risiede nella **semplicità di progettazione** e analisi concettuale.

Teorema 3.1 (Equivalenza DFA-NFA)**.** *Per ogni Automa a Stati Finiti Non-Deterministico (NFA), esiste un Automa a Stati Finiti Deterministico (DFA) equivalente che riconosce lo stesso linguaggio.*

Questo teorema, sebbene non dimostrato qui, è fondamentale: significa che DFAs e NFAs hanno la stessa *potenza espressiva*. Qualsiasi linguaggio riconoscibile da un NFA è riconoscibile anche da un DFA, e viceversa. Questo giustifica l'uso degli NFA per la loro semplicità di programmazione, sapendo che possono essere sempre convertiti in DFAs per l'implementazione pratica.

4 Limitazioni degli Automi a Stati Finiti

Nonostante la loro utilità, gli automi a stati finiti (sia deterministici che non-deterministici) non sono modelli di calcolo universali. Hanno delle limitazioni intrinseche.

Esempio 4.1 (Linguaggio $L = \{a^m b^m \mid m \geq 0\}$)**.** *Consideriamo il linguaggio L sull'alfabeto* $\Sigma = \{a, b\}$ *definito come:*

$$L = \{a^m b^m \mid m \geq 0\}$$

Questo linguaggio include stringhe come ϵ *(per* $m = 0$*), ab, aabb, aaabbb, e così via. Ogni stringa in L è composta da un certo numero di 'a' seguito esattamente dallo stesso numero di 'b'. Un programma Python o Java saprebbe facilmente verificare se una stringa è di questo tipo, ad esempio contando il numero di 'a' e 'b'.*

4.1 Argomento Intuitivo (Pumping Lemma)

Gli automi a stati finiti non sono in grado di riconoscere il linguaggio $L = \{a^m b^m \mid m \geq 0\}$. L'intuizione dietro questa limitazione è che gli automi a stati finiti **non sanno contare** in modo arbitrario e **non hanno memoria esterna** per ricordare conteggi indefiniti.

- **Assunzione per contraddizione**: Supponiamo che esista un DFA D in grado di decidere il linguaggio $L = \{a^m b^m \mid m \geq 0\}$.
- Sia P il numero di stati del DFA D. P è un numero finito e costante.
- **Consideriamo una stringa lunga**: Scegliamo una stringa $s = a^P b^P$. Questa stringa appartiene a L ed ha lunghezza $2P$. Poiché $P \geq 1$ (assumendo almeno uno stato), $2P > P$.
- **Il Principio dei Cassetti (Pigeonhole Principle)**: Quando il DFA D processa i primi P simboli 'a' della stringa s, esso visiterà $P + 1$ configurazioni (partendo da q_0 e leggendo P 'a'). Poiché il DFA ha solo P stati, per il principio dei cassetti, l'automa deve **per forza rivisitare almeno uno stato** durante il processo delle prime P 'a'.
- Questo significa che esiste un **ciclo** nel percorso di computazione dell'automa mentre sta leggendo la porzione a^P della stringa. Possiamo scomporre la stringa s in tre parti: $s = uvw$, dove:
  - u è la parte iniziale della stringa prima che si entri nel ciclo (potrebbe essere vuota).
  - v è la porzione di stringa che corrisponde al ciclo (contiene solo 'a' e non è vuota).
  - w è la parte restante della stringa dopo aver completato il ciclo una volta, che include le 'a' rimanenti (se ci sono) e tutte le 'b'.

  Graficamente: $q_0 \xrightarrow{u} q_i \xrightarrow{v} q_i \xrightarrow{w} q_f \in F$.
- **Implicazione del ciclo**: Se il DFA accetta uvw, dato che v corrisponde a un ciclo, l'automa accetterà anche $uvvw$, $uvvvw$, e in generale $uv^i w$ per qualsiasi $i \geq 0$. Questo perché percorrere il ciclo (v) più volte riporta sempre l'automa nello stesso stato q_i, e da lì la computazione prosegue identica.
- **Contraddizione**: Poiché v contiene solo 'a' (essendo parte dei primi P simboli 'a'), se consideriamo la stringa uv^2w, essa avrà più 'a' di uvw ma lo stesso numero di 'b'. Ad esempio, se $s = a^P b^P$, allora uv^2w sarà della forma $a^{P+|v|}b^P$, dove $|v| > 0$. Questa nuova stringa **non appartiene** al linguaggio L perché il numero di 'a' non è uguale al numero di 'b'.

- Tuttavia, per la proprietà dei cicli, il DFA D dovrebbe accettare uv^2w. Questo è una contraddizione, in quanto D dovrebbe accettare solo stringhe in L e rifiutare quelle non in L.

Conclusione: L'assunzione iniziale che un DFA possa decidere il linguaggio $L = \{a^m b^m \mid m \geq 0\}$ è falsa. Gli automi a stati finiti non hanno la capacità di "contare" in modo arbitrario e confrontare quantità.

Questa limitazione fondamentale degli automi a stati finiti ci spinge a cercare modelli di calcolo più potenti, in grado di svolgere compiti come il conteggio e la memorizzazione di informazioni arbitrarie. Il prossimo passo è l'introduzione delle *Macchine di Turing*, che si dimostreranno essere un modello di calcolo universale, in grado di simulare qualsiasi algoritmo.