Azzolini Riccardo 2020-09-28

Introduzione agli automi a stati finiti

1 Studio dei modelli di computazione come riconoscitori di linguaggi

Tipicamente, si è abituati a vedere la computazione come un processo che conduce da un input a un output, ovvero come il calcolo di una funzione. Il modello di computazione considerato determina allora quali funzioni siano calcolabili.

Tra tutte le possibili funzioni, ci si può concentrare anche solo su quelle che corrispondono al problema del *riconoscimento di un linguaggio* L su un alfabeto Σ. Ciascuna di queste funzioni ha la forma $\Sigma^* \to \{\text{sì}, \text{no}\}$: prende in input una stringa su Σ e determina se essa appartenga o meno al linguaggio L. Concentrarsi solo sul riconoscimento dei linguaggi è sufficiente e vantaggioso perché:

- Limitando l'ambito considerato, diventa più facile trattarlo e formalizzarlo.
- Valutando le classi di linguaggi che diversi modelli di computazione sono in grado di riconoscere, si può confrontare facilmente la loro potenza.
- Il riconoscimento di linguaggi è un aspetto fondamentale dell'uso dei calcolatori:
  - riconoscere un linguaggio di programmazione è il primo passo necessario per trasformare il codice in un programma eseguibile;
  - in generale, tutti i possibili input di un programma costituiscono di fatto un linguaggio, che il programma deve in qualche modo riconoscere.
- Ogni generico problema può essere ridotto al riconoscimento di un linguaggio.

2 Esempio: controller di una porta automatica

Come esempio per introdurre un primo modello di computazione, si consideri una porta automatica che viene attraversata in una sola direzione (dal lato anteriore della porta al lato posteriore). Su ciascun lato della porta è presente un tappetino in grado di rilevare la presenza di una persona. Quando arriva una persona sul rilevatore anteriore, la porta:

1. si apre;
2. lascia attraversare la persona (che passa sul rilevatore posteriore, poi si allontana);

3. si richiude.

Inoltre, supponendo che la porta si apra verso il lato posteriore, per evitare situazioni pericolose essa non deve muoversi (aprirsi o chiudersi) fintanto che è presente una persona sul rilevatore posteriore (dato che questa potrebbe essere colpita dalla porta in movimento).

La porta è gestita da un apposito controller, che riceve in input le informazioni di presenza/assenza di persone sui rilevatori e produce in output un segnale di comando per l'attuatore che apre/chiude la porta. Mettendo insieme le informazioni fornite dai due rilevatori, si può definire l'*alfabeto* di tutti i possibili input:

- front: c'è una persona solo sul rilevatore davanti alla porta;
- rear: c'è una persona solo sul rilevatore dietro alla porta;
- both: ci sono persone su entrambi i rilevatori;
- neither: non ci sono persone sui rilevatori.

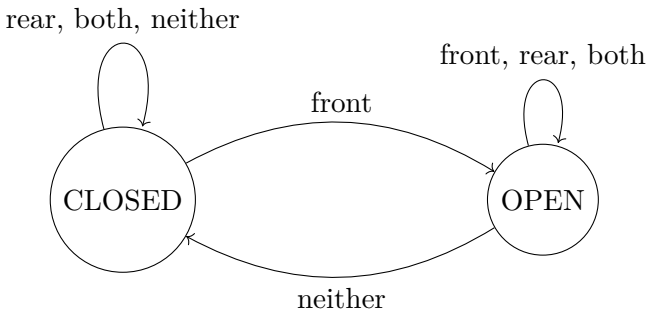
Per poter reagire correttamente a questi input, il controller ha poi bisogno di memorizzare lo **stato** in cui si trova la porta, che può essere CLOSED oppure OPEN. La quantità memoria di cui questo controller ha bisogno è dunque pari a un singolo bit.

In base allo stato corrente e all'input, il controller *cambia stato* (si ha una **transizione**) nel modo descritto dalla seguente *tabella di transizione*:

	neither	front	rear	both
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

Si suppone infine che, quando lo stato cambia, venga inviato all'attuatore della porta il comando corrispondente al nuovo stato (apri/chiudi).

Si osserva che la tabella appena mostrata descrive una funzione che, dati lo stato corrente e l'input, calcola il nuovo stato del controller. Un altro modo equivalente di rappresentare tale funzione è il seguente *diagramma degli stati* (o *diagramma di transizione*):



- uno stato è rappresentato da un cerchio contenente il suo nome;
- una transizione di stato è rappresentata da una freccia etichettata con i simboli degli input che provocano tale transizione;
- gli input che non provocano una transizione sono indicati su un "cappio", una freccia che parte da uno stato e torna allo stesso stato.

Un'altra osservazione importante è che è stato possibile fornire una descrizione finita del comportamento del sistema solo perché sia gli input che gli stati sono finiti. In generale:

- Gli input sono simboli di un alfabeto, quindi sono finiti per la definizione di alfabeto.
- Il vincolo che gli stati siano finiti è la caratteristica principale di questo specifico modello di calcolo, che prende infatti il nome di **automa a stati finiti**. Altri modelli di calcolo non hanno questa limitazione, ma perciò non è possibile descriverli in modo finito.

2.1 Riconoscimento di un linguaggio

L'automa appena presentato può essere studiato come riconoscitore di linguaggi: le sue proprietà possono essere analizzate osservando i linguaggi che esso è in grado di riconoscere. Ad esempio, se si è interessati ad esaminare l'apertura della porta, si può scegliere di considerare il linguaggio formato dalle sequenze di input che, a partire dalla situazione in cui la porta è chiusa, conducono a una situazione in cui la porta è aperta. Formalmente, l'alfabeto (cioè l'insieme dei possibili input) è

$$\Sigma = \{\text{neither}, \text{front}, \text{rear}, \text{both}\}$$

e il linguaggio (su Σ) che si vuole riconoscere è

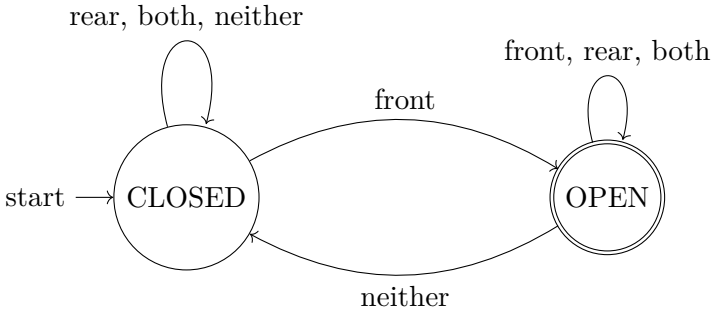
$$L = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ è una sequenza di input che conduce dalla situazione} \\ \text{“porta chiusa” a una situazione “porta aperta”} \end{array} \right\}$$

Per poter utilizzare l'automa come riconoscitore è però necessari aggiungere due informazioni:

- lo **stato iniziale**, che rappresenta la situazione di partenza, ed è indicato nel diagramma degli stati con una freccia entrante priva di sorgente;[1]
- lo **stato finale** o **accettante**, che rappresenta la situazione che si vuole raggiungere, e nel diagramma degli stati viene indicato con un doppio cerchio.

[1] Qui questa freccia è etichettata "start", ma in futuro tale etichetta verrà talvolta omessa, per evitare di aggiungere troppi elementi a diagrammi magari già complessi.

In generale, lo stato iniziale è sempre uno solo, mentre possono esserci anche più stati finali.



Le *stringhe accettate* dall'automa sono tutte le stringhe $w \in \Sigma^*$ che conducono dallo stato iniziale allo stato finale. Ad esempio,

- la stringa "front neither front" è accettata, perché conduce allo stato finale OPEN:

$$\rightarrow \text{CLOSED} \xrightarrow{\text{front}} \text{OPEN} \xrightarrow{\text{neither}} \text{CLOSED} \xrightarrow{\text{front}} \text{OPEN}$$

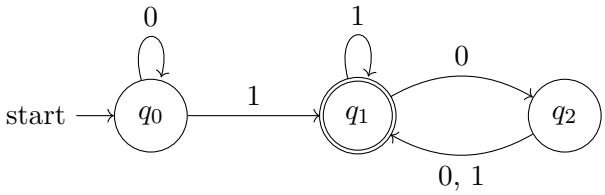
- la stringa "front neither both" non è invece accettata, perché conduce allo stato *non finale* CLOSED:

$$\rightarrow \text{CLOSED} \xrightarrow{\text{front}} \text{OPEN} \xrightarrow{\text{neither}} \text{CLOSED} \xrightarrow{\text{both}} \text{CLOSED}$$

Osservazione: I simboli della stringa di input vengono analizzati uno alla volta, procedendo da sinistra a destra, e ciascuno di essi viene considerato una volta sola: una volta usato, *consumato*, esso non ha più alcun effetto sui comportamenti successivi dell'automa. Di conseguenza, la "lunghezza della computazione" è esattamente uguale alla lunghezza della stringa di input. Questa è un'altra proprietà importante degli automi a stati finiti.

3 Un altro esempio

Come altro esempio, questa volta più astratto, si consideri l'automa descritto dal seguente diagramma degli stati:



Le componenti di questo automa sono:

- gli **stati** $Q = \{q_0, q_1, q_2\}$, che rappresentano la memoria dell'automa (le "situazioni" che può memorizzare);
- le **transizioni**,

  $$q_0 \xrightarrow{0} q_0 \quad q_0 \xrightarrow{1} q_1 \quad q_1 \xrightarrow{0} q_2 \quad q_1 \xrightarrow{1} q_1 \quad q_2 \xrightarrow{0} q_1 \quad q_2 \xrightarrow{1} q_1$$

  che rappresentano le possibili mosse dell'automa — una transizione $p \xrightarrow{a} r$ significa che, se l'automa è nello stato p e il simbolo in input è a, allora l'automa "evolve" nello stato r;
- l'**alfabeto di input** $\Sigma = \{0, 1\}$, l'insieme dei simboli che possono essere forniti in input all'automa;
- lo **stato iniziale** q_0, in cui l'automa si trova all'inizio della "computazione";
- gli **stati finali** o **accettanti** $F = \{q_1\}$.

Data una stringa di simboli in input $w \in \{0, 1\}^*$, l'automa evolve modificando il suo stato in base ai simboli letti, a partire dallo stato iniziale. La *computazione* dell'automa sulla stringa di input w è appunto la sequenza di mosse compiute dall'automa, dove ogni mossa *consuma* un simbolo della stringa di input. Se al termine della computazione, cioè una volta consumati tutti i simboli di w, l'automa si trova in uno stato finale, allora la stringa w è **accettata** (riconosciuta), altrimenti è **rifiutata**. Ad esempio:

- la stringa 001100 viene accettata,

  $$\to q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_1$$

  perché dopo averla consumata l'automa si trova nello stato finale/accettante q_1;
- la stringa 110 viene rifiutata,

  $$\to q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_1 \xrightarrow{0} q_2$$

  perché alla fine l'automa si trova nello stato non accettante q_2.