Automi a stati finiti non deterministici: definizioni formali

Materia	Fondamenti di Informatica: Linguaggi Formali
Data	@April 15, 2024

Linguaggio riconosciuto

Per definire il linguaggio, dobbiamo ricordare che una configurazione può ammettere più di una configurazione successiva. La computazione di un automa a stati finiti non deterministico è quindi **un albero**, in cui ogni nodo è una configurazione e come figli ha tutte le possibili configurazioni. Una stringa verrà riconosciuta **solamente se esiste una configurazione finale nelle foglie dell'albero di computazione**.

Dobbiamo quindi utilizzare il concetto di **configurazione successiva** nel caso non deterministico.

Configurazioni successive

Le configurazioni successive ad una configurazione (q, ax) possono essere rappresentate con $(\{ q_0, q_1 ..., q_n \}, x)$. Le configurazioni successive ovviamente si differiscono solamente nello stato, ma non nella stringa da leggere. Possiamo quindi utilizzare sempre il simbolo $\vdash$.

Possiamo estendere la nozione di configurazione successiva considerando anche **tutte le configurazioni successive a partire da un insieme di configurazioni**. In un passo quindi passeremo da un insieme di configurazioni ad un altro insieme di configurazioni. Quindi si passa da $(\{ q_0, ..., q_n \}, ax) \vdash (\{ q_i, ..., q_j \}, x)$.

Così come abbiamo definito la relazione $\vdash$ nel caso deterministico, anche in questo caso possiamo definire la **chiusura transitiva** $\vdash^+$. Siano I_1, I_2 due insiemi di configurazioni. Allora scriveremo $I_1 \vdash^+ I_2$ se possiamo passare con $n \geq 1$

passi dall'insieme di configurazioni I_1 all'insieme di configurazioni I_2. Allo stesso modo, possiamo definire la **chiusura riflessiva** $\vdash^*$**,** ossia se possiamo passare da I_1 a I_2 in 0 o più passi.

Ora possiamo definire il linguaggio riconosciuto. Diremo che una stringa è riconosciuta dall'automa se dalla configurazione iniziale possiamo passare ad un insieme di configurazioni in cui **esiste almeno** 1 **configurazione finale:**

$$x \in L(\mathbb{A}_N) \iff (\{ q_0 \}, x) \vdash^* (I, \epsilon) : I \subseteq Q \land I \cap F \neq \emptyset$$

Quindi sia $\mathbb{A}_N =< Q, \Sigma, \delta_N, q_0, F >$ un ASFND. Il **linguaggio riconosciuto** da $\mathbb{A}_n$, indicato con $L(\mathbb{A}_N)$, è così definito:

$$L(\mathbb{A}_N) = \{ x \in \Sigma^* : (q_0, x) \vdash^* (I, \epsilon) \; con \; I \subseteq Q \land I \cap F \neq \emptyset \}$$

Funzione di transizione estesa

Anche in questo caso definiamo la **funzione di transizione estesa:**

$$\overline{\delta} : Q \times \Sigma^* \to \mathrm{P}(Q)$$

Possiamo definire la funzione $\overline{\delta}_N$ allo stesso modo con cui abbiamo definito la funzione $\overline{\delta}$ per l'automa a stato finito deterministico. L'unico problema nella definizione tuttavia è che la funzione δ_N, così come la funzione $\overline{\delta}_N$, ha come codominio l'insieme $\mathrm{P}(Q)$, ossia **sottoinsiemi di stati**, e naturalmente non possiamo utilizzarlo come parametro per la funzione δ_N.

Possiamo rimediare con la seguente definizione. La funzione di transizione estesa $\overline{\delta}_N$ in un ASFND è definita nel seguente modo:

- $\overline{\delta}_N(q, \epsilon) = \{ q \}$;
- $\overline{\delta}_N(q, xa) = \bigcup_{p \in \overline{\delta}_N(q,x)} \delta_N(p, a)$.

Ossia consideriamo **ogni stato dell'insieme che si ottiene come immagine di** $\overline{\delta}_N(q, x)$ **e calcoliamo** $\delta_N(p, a)$.

In realtà possiamo anche dare la definizione di $\overline{\delta}_N(q, ax)$, così come per l'automa a stato finito deterministico.

A questo punto possiamo definire il linguaggio riconosciuto dall'automa $\mathbb{A}_N$ in virtù della funzione $\overline{\delta}_N$:

$$L(\mathbb{A}_N) = \{\, x \in \Sigma^* : \overline{\delta}_N(q_0, x) \cap F \neq \emptyset \,\}$$

Dovremo dimostrare che vale la seguente proposizione:

$$\overline{\delta}_N(q_0, x) \cap F \neq \emptyset \iff (\{\, q_0 \,\}, x) \vdash^* (I, \epsilon) : \; I \cap F \neq \emptyset$$

Per dire che entrambe le definizioni di linguaggio riconosciuto sono equivalenti, ossia che gli insiemi coincidono. Si dimostra per **induzione sulla lunghezza di** x.

Costruzione dei sottoinsiemi

La **costruzione dei sottoinsiemi** è la procedura algoritmica che permette di costruire un automa a stati finito deterministico a partire da un automa a stati finito non deterministico. L'insieme degli stati totali dell'automa a stati finiti deterministico ha cardinalità 2^n, dove n è il numero degli stati dell'automa a stati finiti non deterministico Q: questo perché dovremmo passare da un sottoinsieme ad un altro per ogni step della computazione. Ci potrebbero essere inoltre degli stati non accessibili.

Ci sono degli automi non deterministici tali che il corrispondente automa deterministico ha 2^n stati totali, quindi il limite superiore viene raggiunto.

Linguaggio regolare

Un linguaggio si dice **regolare** se esiste un automa a stati finiti che lo riconosce. La classe dei linguaggi regolari gode di numerose proprietà e diverse caratterizzazioni (ossia un linguaggio è regolare se e solo se vale una determinata proprietà). L'importanza di avere numerose proprietà è che se viene dimostrata una proprietà per un certo strumento (i.e. grammatica), allora tale proprietà si estende alla classe.

Espressioni regolari

Caratteristica dei linguaggi regolari sono le **espressioni regolari**. Le espressioni regolari permettono di rappresentare certi linguaggi.