Azzolini Riccardo 2020-10-27

Costruzione per sottoinsiemi

1 Dimostrazione di equivalenza

Dato un NFA $N = \langle Q, \Sigma, \delta, q_0, F \rangle$, si è visto che il DFA definito dalla costruzione per sottoinsiemi (considerando, per semplicità, tutti i possibili sottoinsiemi di Q, cioè senza valutazione differita) è $D_N = \langle Q_D, \Sigma, \delta_D, \{q_0\}, F_D \rangle$, dove:

$$Q_D = 2^Q$$
$$F_D = \{S \in Q_D \mid S \cap F \neq \varnothing\}$$
$$\forall S \subseteq Q,\ \forall a \in \Sigma \quad \delta_D(S, a) = \bigcup_{p \in S} \delta(p, a)$$

Adesso, si vuole dimostrare che D_N e N riconoscono lo stesso linguaggio. Analogamente a quanto fatto nel verso opposto (da un DFA a un NFA), si dimostra prima un lemma che mette in relazione le computazioni dei due automi, e poi questo viene usato per mostrare l'equivalenza:

- *Lemma*: Dato un NFA $N = \langle Q, \Sigma, \delta, q_0, F \rangle$, per ogni $w \in \Sigma^*$ si ha che $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}(q_0, w)$.
- *Teorema*: Dato un NFA $N = \langle Q, \Sigma, \delta, q_0, F \rangle$, $L(D_N) = L(N)$.

Sull'enunciato del lemma può essere fatta un'osservazione. $\hat{\delta}(q_0, w)$ è una computazione di un NFA, quindi restituisce un insieme di stati dell'NFA: $\hat{\delta}(q_0, w) \in 2^Q$. Invece, $\hat{\delta}_D(\{q_0\}, w)$ è una computazione di un DFA, che restituisce un singolo stato del DFA, $\hat{\delta}_D(\{q_0\}, w) \in Q_D$, ma l'insieme degli stati di tale automa è stato scelto proprio in modo che $Q_D = 2^Q$. Perciò, l'uguaglianza $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}(q_0, w)$ ha senso.

1.1 Dimostrazione del lemma

La dimostrazione del lemma procede per induzione su $|w|$.

- *Base*: $|w| = 0$, cioè $w = \epsilon$. Per dimostrare questo caso è sufficiente applicare le definizioni delle due funzioni di transizione estese:

  $$\hat{\delta}_D(\{q_0\}, \epsilon) = \{q_0\} \qquad \hat{\delta}(q_0, \epsilon) = \{q_0\}$$

  quindi $\hat{\delta}_D(\{q_0\}, \epsilon) = \hat{\delta}(q_0, \epsilon)$.

- *Passo induttivo*: $|w| > 0$, cioè $w = xa$, con $x \in \Sigma^*$ e $a \in \Sigma$. Per ipotesi induttiva (IH), si suppone che $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}(q_0, x)$. Allora, partendo dal lato sinistro dell'uguaglianza da dimostrare:

$$\begin{aligned}
\hat{\delta}_D(\{q_0\}, xa) &= \delta_D(\hat{\delta}_D(\{q_0\}, x), a) && \text{[per definizione di } \hat{\delta}_D\text{]} \\
&= \bigcup_{p \in \hat{\delta}_D(\{q_0\}, x)} \delta(p, a) && \text{[per definizione di } \delta_D\text{]} \\
&= \bigcup_{p \in \hat{\delta}(q_0, x)} \delta(p, a) && \text{[per (IH)]} \\
&= \hat{\delta}(q_0, xa) && \text{[per definizione di } \hat{\delta}\text{]}
\end{aligned}$$

Avendo dimostrato sia la base che il passo induttivo, il lemma è dimostrato.

1.2 Dimostrazione del teorema

Si dimostra $L(D_N) = L(N)$ facendo vedere che, per ogni $w \in \Sigma^*$, $w \in L(N)$ se e solo se $w \in L(D_N)$:

$$\begin{aligned}
w \in L(N) &\iff \hat{\delta}(q_0, w) \cap F \neq \varnothing && \text{[per definizione di stringa accettata da un NFA]} \\
&\iff \hat{\delta}_D(\{q_0\}, w) \cap F \neq \varnothing && \text{[per il lemma]} \\
&\iff \hat{\delta}_D(\{q_0\}, w) \in F_D && \text{[per definizione di } F_D\text{]} \\
&\iff w \in L(D_N) && \text{[per definizione di stringa accettata da un DFA]}
\end{aligned}$$

2 Esempio di caso sfavorevole

Si consideri il linguaggio

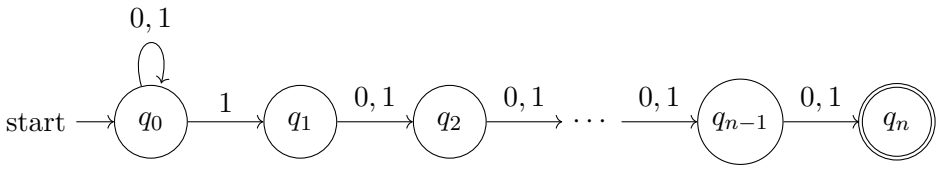
$$L^n = \{w \in \{0,1\}^* \mid |w| \geq n \text{ e l'}n\text{-esimo simbolo dalla fine è } 1\}$$

definito in dipendenza di un parametro $n \geq 1$. Ogni stringa $w \in L^n$ ha la struttura

$$w = a_1 \dots a_k \mathbf{1} b_1 \dots b_{n-1}$$

dove $k \geq 0$ e $a_i, b_j \in \{0,1\}$.

Indipendentemente dal valore di n, esiste un NFA N^n che riconosce il linguaggio L^n, cioè tale che $L(N^n) = N^n$. Questo automa, che ha $n+1$ stati ($Q = \{q_0, \dots, q_n\}$), è rappresentato dal seguente diagramma:



Data in input una stringa $w \in L^n$, una delle possibili computazioni di questo automa è quella che: la computazione di questo automa può:

1. rimane nello stato q_0, sfruttando il cappio, per leggere i simboli $a_1 \dots a_k$;
2. passa da q_0 a q_1 quando viene letto il simbolo 1 situato a n posizioni dalla fine della stringa;
3. passa infine da q_1 a q_n consumando i simboli $b_1 \dots b_{n-1}$.

Tale computazione termina nello stato finale q_n, dunque la stringa viene accettata.

Data invece una stringa $w \notin L^n$, cioè che non ha un 1 a n posizioni dalla fine, ci sono tre possibilità:

- w è formata solo da simboli 0, quindi l'automa rimane sempre solo nello stato q_0;
- w contiene un 1 seguito da troppi pochi simboli, quindi non si raggiunge lo stato finale q_n;
- w contiene un 1 seguito da troppi simboli, quindi la computazione si blocca perché lo stato finale q_n non ha transizioni uscenti.

In tutti questi casi, la stringa non è accettata.

Queste osservazioni dimostrano in modo informale che il linguaggio $L(N^n)$ riconosciuto da N^n è effettivamente L^n, qualunque sia il valore del parametro n (purché tale valore sia scelto tra quelli ammessi, $n \geq 1$).

Si osserva che il numero di stati dell'automa, $n + 1$, è linearmente correlato con il parametro n che caratterizza il linguaggio riconosciuto. Ciò significa, in qualche modo, che la dimensione[1] dell'automa è "ragionevole" rispetto al parametro che caratterizza il problema considerato (il riconoscimento del linguaggio L^n).[2] In seguito, si dimostrerà che la stessa dimensione ragionevole non può essere ottenuta con un DFA:

[1] La dimensione di un automa a stati finiti (deterministico o non deterministico) è data, sostanzialmente, dalla dimensione $|\Sigma|$ dell'alfabeto in input e dal numero $|Q|$ di stati (che, insieme, determinano la dimensione della tabella di transizione). Di questi due parametri, però, $|\Sigma|$ è fissato in base all'alfabeto del linguaggio da riconoscere, quindi solo $|Q|$ può variare tra diversi automi che riconoscono lo stesso linguaggio. Dunque, in generale, il numero di stati di un automa costituisce un buon parametro per valutarne la dimensione / complessità.

[2] Il parametro n è rilevante per la caratterizzazione della complessità del problema del riconoscimento di L^n in quanto corrisponde alla lunghezza minima di una stringa $w \in L^n$: siccome un automa deve leggere l'intera stringa per poterla accettare, il riconoscimento di w richiede almeno n passi di computazione.

Teorema: Non esiste alcun DFA con meno di 2^n stati che riconosca L^n.

Questo teorema implica che:

- In alcuni casi, la costruzione per sottoinsiemi genera DFA che sono inevitabilmente grandi rispetto all'NFA di partenza. Ad esempio, in questo caso, il DFA generato (senza valutazione differita dei sottoinsiemi) ha $|Q_D| = |2^Q| = 2^{|Q|} = 2^{n+1}$ stati, e "più o meno" tutti questi stati servono: dopo aver tolto gli stati irrilevanti, per il teorema devono rimanere ancora almeno 2^n stati.
- Ci sono problemi (linguaggi da riconoscere) per cui esistono NFA piccoli (aventi un numero di stati "proporzionato" al problema), ma che non ammettono invece DFA piccoli. Infatti, il teorema non si riferisce in particolare alla costruzione per sottoinsiemi, ma piuttosto considera in generale l'esistenza di un DFA che possa riconoscere il linguaggio L^n.

2.1 Dimostrazione

Il teorema viene dimostrato ragionando per assurdo, cioè supponendo che esista un DFA $D = \langle Q, \{0,1\}, \delta, q_0, F\rangle$ tale che $|Q| < 2^n$ e $L(D) = L^n$.

Si consideri l'insieme

$$S = \{w \in \{0,1\}^* \mid |w| = n\}$$

cioè l'insieme delle stringhe su $\{0,1\}$ la cui lunghezza è esattamente n. S ha due proprietà:

1. $|S| = 2^n$ (esistono 2^n diverse stringhe di lunghezza n su un alfabeto di due simboli).
2. Esistono due stringhe $w_1, w_2 \in S$ tali che $w_1 \neq w_2$ e $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$.

Per dimostrare la proprietà 2, si suppone che essa non valga, ovvero che, per ogni $w_1, w_2 \in S$ tali che $w_1 \neq w_2$, sia $\hat{\delta}(q_0, w_1) \neq \hat{\delta}(q_0, w_2)$. In altre parole, la computazione dell'automa D a partire da q_0 porterebbe a uno stato diverso per ogni stringa in S, quindi gli stati di D dovrebbero essere almeno tanti quanti le stringhe in S, cioè 2^n, contrariamente all'ipotesi $|Q| < 2^n$. Allora, la proprietà 2 è verificata.

Adesso, si considerano due stringhe $w_1, w_2 \in S$,

$$w_1 = a_1 \dots a_n \quad w_2 = b_1 \dots b_n \qquad \text{con } a_i, b_j \in \{0,1\}$$

scelte in modo che soddisfino la proprietà 2: $w_1 \neq w_2$ e $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. Essendo due stringhe diverse, esiste almeno una posizione in cui queste differiscono: $\exists j \in \{1, \dots, n\}$ tale che $a_j \neq b_j$. Sia $k \in \{1, \dots, n\}$ il più piccolo indice (la prima delle una o più posizioni) in cui le stringhe differiscono ($a_k \neq b_k$). Per dimostrare il teorema si ragionerà sul valore di k, considerando in particolare le situazioni $k = 1$ e $k > 1$, e mostrando che in entrambi questi casi si giunge a un assurdo.

Non avendo altre posizioni in cui le stringhe potrebbero differire, deve esserci qualcosa di non valido nelle assunzioni fatte prima di dimostrare la proprietà 2, che erano $L(D) = L^n$ e $|Q| \geq 2^n$. Segue che un DFA con queste proprietà non può esistere: il teorema è dimostrato.

2.1.1 Caso $k = 1$

Se $k = 1$, significa che le due stringhe iniziano con simboli diversi. Qui si sceglie di considerare il caso

$$w_1 = \mathbf{0}\, a_2 \ldots a_n$$
$$w_2 = \mathbf{1}\, b_2 \ldots b_n$$

(per il caso duale, in cui w_1 inizia con 1 e w_2 con 0, la dimostrazione è simmetrica).

Per ipotesi, il linguaggio riconosciuto dall'automa D è $L(D) = L^n$, quello che comprende le stringhe aventi un 1 a n posizioni dalla fine. Siccome $w_2 \in L^n$, mentre $w_1 \notin L^n$, si ha che D accetta w_2 e rifiuta w_1. Allora, per la definizione di stringa accettata da un DFA, $\hat{\delta}(q_0, w_2)$ deve essere uno stato finale, e $\hat{\delta}(q_0, w_1)$ deve essere non finale.

w_1 e w_2 sono però state scelte in modo che soddisfacessero la proprietà 2, per cui le due computazioni portano allo stesso stato, $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$, e tale stato:

- o è finale, quindi entrambe le stringhe vengono accettate;
- o è non finale, quindi entrambe le stringhe vengono rifiutate.

Il fatto che una stringa venga accettata e l'altra no è dunque *assurdo*: da ciò si deduce che w_1 e w_2 non possono differire nella posizione $k = 1$.

2.1.2 Caso $k > 1$

Avendo escluso la possibilità che le due stringhe differiscano in posizione $k = 1$, si suppone che invece sia $k > 1$. Allora, si ha:[3]

$$w_1 = a_1 \ldots a_{k-1}\, \mathbf{0}\, a_{k+1} \ldots a_n$$
$$w_2 = b_1 \ldots b_{k-1}\, \mathbf{1}\, b_{k+1} \ldots b_n$$

(anche qui, il caso in cui $a_k = 0$ e $b_k = 1$ è simmetrico).

A partire da w_1 e w_2, si costruiscono due nuove stringhe w_1' e w_2', concatenando una sequenza di $k - 1$ zeri, indicata con 0^{k-1}, alla fine delle stringhe date:

$$w_1' = w_1 0^{k-1} = a_1 \ldots a_{k-1}\, 0\, a_{k+1} \ldots a_n\, 0^{k-1}$$
$$w_2' = w_2 0^{k-1} = b_1 \ldots b_{k-1}\, 1\, b_{k+1} \ldots b_n\, 0^{k-1}$$

[3] Per la precisione, siccome k è stata definita come la prima posizione in cui w_1 e w_2 differiscono, si deve avere $a_1 \ldots a_{k-1} = b_1 \ldots b_{k-1}$. Invece, tra i simboli successivi alla k-esima posizione potrebbero esserci altre differenze.

Per entrambe queste stringhe, la k-esima posizione si trova a n posizioni dalla fine. Di conseguenza, $w_1' \in L^n$ e $w_2' \notin L^n$, cioè, per ipotesi, D accetta w_1' e rifiuta w_2'.

Considerando adesso le computazioni dell'automa, poiché w_1 e w_2 sono scelte in modo da soddisfare la proprietà 2, si ha che $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. La computazione sulla stringa w_1' è poi data dalla computazione su w_1 seguita da quella su 0^{k-1}, e lo stesso vale per la computazione su w_2'. Dunque, complessivamente, le computazioni su w_1' e w_2' portano allo stesso stato:

$$\begin{aligned}
\hat{\delta}(q_0, w_1') &= \hat{\delta}(q_0, w_1 0^{k-1}) \\
&= \hat{\delta}(\hat{\delta}(q_0, w_1), 0^{k-1}) \\
&= \hat{\delta}(\hat{\delta}(q_0, w_2), 0^{k-1}) && \text{[per la proprietà 2]} \\
&= \hat{\delta}(q_0, w_2 0^{k-1}) \\
&= \hat{\delta}(q_0, w_2')
\end{aligned}$$

Di nuovo, ci si trova in una situazione in cui le computazioni su due stringhe portano a uno stesso stato, ma una viene accettata e l'altra no, che è *assurdo*, perché lo stato risultante o è finale o non lo è: w_1' e w_2' possono solo essere o entrambe accettate, o entrambe rifiutate da D.

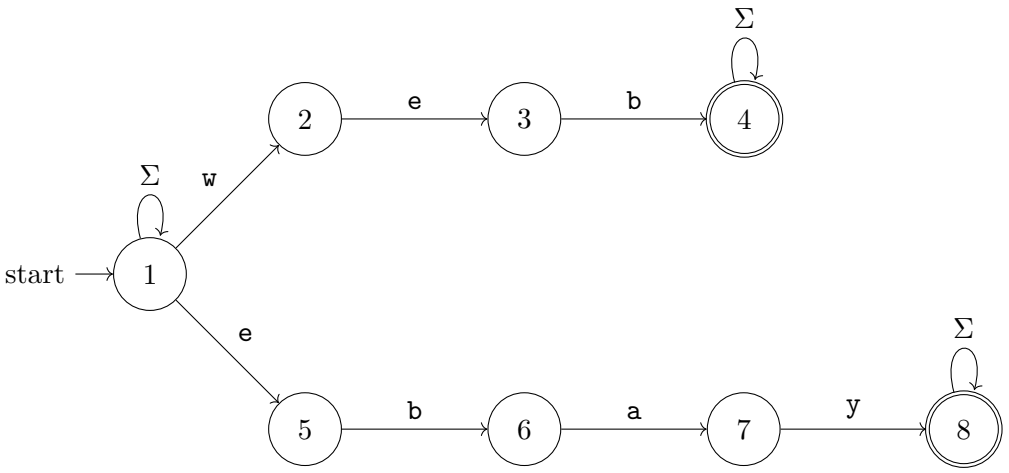
3 Esempio: ricerca di parole chiave

Una tipica applicazione degli automi a stati finiti è la ricerca di parole chiave all'interno di dei documenti. Sia ad esempio $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{x}, \mathtt{y}, \mathtt{z}\}$ l'alfabeto inglese. Se si pensa ai documenti come stringhe $w \in \Sigma^*$, il linguaggio

$$L = \{w \in \Sigma^* \mid w = \alpha\mathtt{web}\beta \text{ o } w = \alpha\mathtt{ebay}\beta \text{ con } \alpha, \beta \in \Sigma^*\}$$

è l'insieme dei documenti che contengono le parole chiave `web` o `ebay`.

Questo linguaggio può essere riconosciuto da un NFA piuttosto semplice:



Applicando a esso la costruzione per sottoinsiemi, e considerando solo gli stati raggiungibili, si ottiene un DFA che ha lo stesso numero di stati di questo NFA. Le transizioni del DFA sono invece molte di più di quelle dell'NFA, dato che, per definizione, il DFA deve specificare una transizione uscente da ciascuno stato per ogni simbolo dell'alfabeto di input. Dunque, se da un lato il DFA è più semplice da implementare, e formalmente ha una complessità ancora proporzionata al problema (avendo lo stesso numero di stati dell'NFA), dall'altro lato esso è ben più difficile da leggere.

In generale, per il problema della ricerca delle parole chiave, il numero di stati del DFA risultante dalla costruzione per sottoinsiemi *non è mai maggiore* del numero di stati dell'NFA di partenza. Questa garanzia rende pratico l'uso della costruzione per sottoinsiemi nell'implementazione di programmi per la ricerca di parole chiave.