Azzolini Riccardo 2020-11-09

Dalle espressioni regolari agli ϵ-NFA

1 Dalle espressioni regolari agli ϵ-NFA

In precedenza, si è affermato che ogni linguaggio regolare è riconosciuto da un'espressione regolare. Il verso opposto, cioè il fatto che sia regolare ogni linguaggio riconosciuto da un'espressione regolare, è dato dal seguente teorema:

Teorema: Data un'espressione regolare E, esiste un ϵ-NFA A_E che riconosce il linguaggio generato da E, cioè tale che $L(A_E) = L(E)$.

Si sceglie di generare un ϵ-NFA perché, come si vedrà in seguito, la corrispondenza tra le espressioni regolari e questo tipo di automi è molto semplice. Cercare invece di passare direttamente a un NFA, o soprattutto a un DFA, risulterebbe notevolmente più complicato.

2 Algoritmo di McNaughton-Yamada-Thompson

La dimostrazione (costruttiva) del teorema appena enunciato è data dall'*algoritmo di McNaughton-Yamada-Thompson*, che appunto riceve un input un'espressione regolare E e produce in output un ϵ-NFA A_E tale che $L(A_E) = L(E)$.

Quest'algoritmo definisce l'automa A_E operando induttivamente[1] sulla definizione delle espressioni regolari: quando E è un'espressione contenente un operatore, vengono prima costruiti gli automi corrispondenti alle sue "sottoespressioni" (gli argomenti dell'operatore), e poi, a partire da questi, si costruisce A_E. L'induzione sulla definizione delle espressioni regolari prevede:

- tre casi base, corrispondenti alle espressioni ϵ, $\varnothing$ e $a \in \Sigma$;
- quattro casi induttivi, corrispondenti agli operatori $+$, $\cdot$, * e $()$.

Ogni automa A costruito a ciascun passo dall'algoritmo soddisfa le seguenti proprietà,[2] necessarie per dimostrare formalmente che $L(A_E) = L(E)$:

- (P1) A ha un unico stato finale;

[1] A livello implementativo, l'induzione corrisponde naturalmente alla ricorsione, ma in molti casi non è troppo difficile trasformarla in un'iterazione.

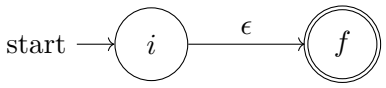
[2] Siccome devono essere soddisfatte a ogni passo, si dice che queste proprietà sono un'*invariante* dell'algoritmo.

- (P2) A non ha transizioni uscenti dallo stato finale;
- (P3) A non ha transizioni entranti nello stato iniziale.

Nella maggior parte dei passi, l'algoritmo introduce due stati *nuovi*, i e f. Essi sono nuovi nel senso che si assume siano distinti da quelli introdotti nei passi precedenti (in un'implementazione, ciò corrisponderebbe a generare ogni volta dei nomi diversi).

2.1 Passo base $E = \epsilon$

L'automa A_ϵ, corrispondente all'espressione regolare ϵ, è quello descritto dal seguente diagramma di transizione:



Formalmente, $A_\epsilon = \langle \{i, f\}, \{a\}, \delta, i, \{f\} \rangle$, dove la funzione di transizione è definita come:

$$\begin{aligned} \delta(i, \epsilon) &= \{f\} & \delta(i, a) &= \varnothing \\ \delta(f, \epsilon) &= \varnothing & \delta(f, a) &= \varnothing \end{aligned}$$

Si osserva che, per definizione, un automa deve sempre avere un alfabeto, e quest'ultimo è un insieme non vuoto, quindi è necessario specificare almeno un simbolo a. In questo caso tale simbolo "non viene usato", dunque può essere scelto arbitrariamente. Se A_ϵ viene definito nell'ambito della costruzione dell'automa per un'espressione più complessa, si può usare un simbolo presente altrove nell'espressione, che prima o poi verrebbe "incontrato" comunque (evitando così la presenza di un simbolo "inutile" nell'alfabeto).

Dato l'automa A_ϵ, è immediato verificare intuitivamente che esso soddisfa le proprietà (P1)–(P3): si vede direttamente dal diagramma che esso ha un solo stato finale f, nessuna transizione uscente da esso e nessuna transizione entrante nello stato iniziale i.

Anche il fatto che $L(A_\epsilon) = \{\epsilon\} = L(\epsilon)$ è abbastanza intuitivo:

- in assenza di simboli in input, l'automa passa da i allo stato finale f grazie all'ϵ-transizione, perciò la stringa vuota viene accettata;
- qualunque simbolo in input fa bloccare tutte le possibili computazioni, quindi ogni stringa non vuota viene rifiutata.

2.2 Passo base $E = \varnothing$

L'automa $A_\varnothing$ è descritto dal diagramma di transizione

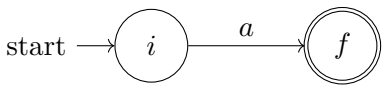
cioè, formalmente, è $A_\varnothing = \langle \{i, f\}, \{a\}, \delta, i, \{f\} \rangle$, dove a è un simbolo qualunque (come nel caso $E = \epsilon$), mentre la funzione di transizione è definita come:

$$\forall q \in \{i, f\},\ \forall s \in \{a, \epsilon\} \quad \delta(q, s) = \varnothing$$

Anche qui è immediato verificare che $A_\varnothing$ soddisfa (P1)–(P3). Si ha poi che $L(A_\varnothing) = \varnothing = L(\varnothing)$ perché, intuitivamente, l'automa non ha modo di raggiungere il suo stato finale, dunque non può accettare alcuna stringa.

2.3 Passo base $E = a$

L'automa A_a costruito in questo caso è



ovvero $A_a = \langle \{i, f\}, \{a\}, \delta, i, \{f\} \rangle$, con:

$$\begin{aligned} \delta(i, \epsilon) &= \varnothing \qquad \delta(i, a) = \{f\} \\ \delta(f, \epsilon) &= \varnothing \qquad \delta(f, a) = \varnothing \end{aligned}$$

Si verifica immediatamente che A_a soddisfa (P1)–(P3) e, mediante un ragionamento simile a quello fatto nel caso $E = \epsilon$, che $L(A_a) = \{a\} = L(a)$.

2.4 Passo induttivo $E = R + S$

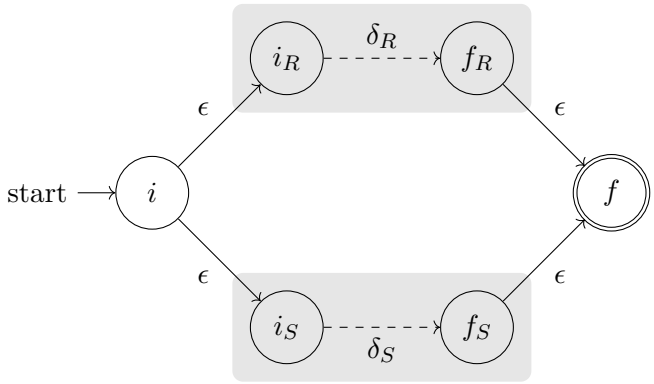
Per ipotesi induttiva, dalle sottoespressioni R e S vengono costruiti due automi

$$A_R = \langle Q_R, \Sigma_R, \delta_R, i_R, \{f_R\} \rangle \qquad A_S = \langle Q_S, \Sigma_S, \delta_S, i_S, \{f_S\} \rangle$$

che soddisfano (P1)–(P3) e sono tali che $L(A_R) = L(R)$ e $L(A_S) = L(S)$.

Per fare in modo che questi due automi rimangano "separati" anche se inseriti come parti di un singolo automa più grande, bisogna imporre che essi non abbiano nomi di stati in comune: deve essere $Q_R \cap Q_S = \varnothing$. Tale condizione può comunque sempre essere garantita, eventualmente rinominando gli stati di uno dei due automi (e in un'implementazione sarebbe sufficiente generare, per ogni nuovo stato introdotto, un nome diverso da quelli di tutti gli stati presenti negli automi ottenuti ai passaggi precedenti).

Dati A_R e A_S, l'automa A_{R+S} è costruito come segue:



Esso comprende due "blocchi" che simulano il comportamento degli automi A_R e A_S, cioè all'interno dei quali la funzione di transizione δ del nuovo automa segue i comportamenti di δ_R e δ_S, permettendo gli stessi percorsi di computazione. Si osserva però che, in questi blocchi, gli stati provenienti da A_R e A_S perdono i loro ruoli di stati iniziali e finali, che vengono invece assunti dai nuovi stati i e f.

2.4.1 Definizione formale dell'automa

Formalmente, l'automa considerato è definito come:

$$A_{R+S} = \langle Q_R \cup Q_S \cup \{i, f\}, \Sigma_R \cup \Sigma_S, \delta, i, \{f\} \rangle$$

In particolare, la funzione di transizione δ può essere intuitivamente compresa dal diagramma di transizione, ma la sua definizione formale è un po' complicata da scrivere. Infatti, per chi fosse interessato, è necessario:

- definire le transizioni uscenti dai nuovi stati, i e f:

$$\begin{aligned} & \delta(i, \epsilon) = \{i_R, i_S\} \\ \forall s \in \Sigma_R \cup \Sigma_S \quad & \delta(i, s) = \varnothing \\ \forall s \in \Sigma_R \cup \Sigma_S \cup \{\epsilon\} \quad & \delta(f, s) = \varnothing \end{aligned}$$

- definire le transizioni uscenti da quelli che erano gli stati finali di A_R e A_S:

$$\begin{aligned} & \delta(f_R, \epsilon) = \delta(f_S, \epsilon) = \{f\} \\ \forall s \in \Sigma_R \cup \Sigma_S \quad & \delta(f_R, s) = \delta(f_S, s) = \varnothing \end{aligned}$$

- simulare il comportamento di A_R sui simboli presenti nell'alfabeto Σ_R (tranne per lo stato f_R, il cui comportamento cambia, data l'aggiunta dell'ϵ-transizione verso f),

$$\forall q \in Q_R \setminus \{f_R\},\ \forall s \in \Sigma_R \cup \{\epsilon\} \quad \delta(q, s) = \delta_R(q, s)$$

  specificando che invece gli stati di A_R non hanno transizioni etichettate con i simboli presenti solo nell'alfabeto Σ_S:

$$\forall q \in Q_R \setminus \{f_R\},\ \forall s \in \Sigma_S \setminus \Sigma_R \quad \delta(q, s) = \varnothing$$

- analogamente, simulare il comportamento di A_S:

$$\begin{aligned} \forall q \in Q_S \setminus \{f_S\},\ \forall s \in \Sigma_S \cup \{\epsilon\} \quad & \delta(q, s) = \delta_S(q, s) \\ \forall q \in Q_S \setminus \{f_S\},\ \forall s \in \Sigma_S \setminus \Sigma_R \quad & \delta(q, s) = \varnothing \end{aligned}$$

In pratica, per fare dimostrazioni semi-formali, è sufficiente riferirsi al diagramma di transizione.

2.4.2 Dimostrazione di correttezza

Ancora una volta, è immediato verificare che A_{R+S} soddisfa (P1)–(P3). Invece, per dimostrare in modo semi-formale che $L(A_{R+S}) = L(R) \cup L(S) = L(R + S)$, è necessario un attimo di ragionamento in più rispetto ai casi base.

Innanzitutto, per ipotesi induttiva si ha che $L(A_R) = L(R)$ e $L(A_S) = L(S)$. È allora sufficiente dimostrare che $L(A_{R+S}) = L(A_R) \cup L(A_S)$, ovvero che, data una stringa $w \in (\Sigma_R \cup \Sigma_S)^*$,

$$\begin{aligned} w \in L(A_{R+S}) & \iff w \in L(A_R) \cup L(A_S) \\ & \iff w \in L(A_R) \text{ o } w \in L(A_S) \end{aligned}$$

Come al solito, si trattano separatamente i due versi del "se e solo se".

- $\implies$: $w \in L(A_{R+S})$ significa che almeno una delle computazioni di A_{R+S} per la alla stringa w definisce un percorso che porta dallo stato i allo stato f. Intuitivamente, guardando il diagramma di transizione, questo percorso o passa "da sopra" oppure passa "da sotto".

  Si consideri ad esempio il caso in cui si passa "da sopra". La prima e l'ultima transizione del percorso sono ϵ-transizioni, quindi la stringa deve essere consumata interamente nel passaggio da i_R a f_R. Esiste dunque un percorso da i_R a f_R, definito da δ_R, che consuma esattamente la stringa w. Tale percorso era presente anche in A_R, l'automa che il blocco da i_R a f_R simula, mantenendo in particolare il comportamento della funzione di transizione δ_R. Segue che w è accettata anche da A_R: $w \in L(A_R)$.

  Nel caso del passaggio "da sotto" si procede in modo analogo, arrivando a dedurre che $w \in L(A_S)$. È allora dimostrato che

$$w \in L(A_{R+S}) \implies w \in L(A_R) \cup L(A_S)$$

- $\impliedby$: Se $w \in L(A_R) \cup L(A_S)$, deve esistere una percorso che, consumando w, va da i_R a f_R in base a δ_R, oppure va da i_S a f_S in base a δ_S. In entrambi i casi, questo percorso può essere esteso, aggiungendo un'ϵ-transizione iniziale e una finale, per ottenere un percorso da i a f, quindi $w \in L(A_{R+S})$. Così, è dimostrato anche che

$$w \in L(A_{R+S}) \impliedby w \in L(A_R) \cup L(A_S)$$

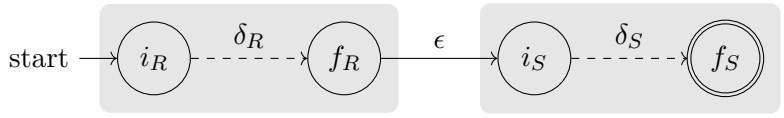
Per una dimostrazione più formale, bisognerebbe lavorare sulle funzioni di transizione estese, mostrando che il comportamento della $\hat{\delta}$ di A_{R+S} può essere ricondotto a quello di $\hat{\delta}_R$ e $\hat{\delta}_S$. Questa dimostrazione è abbastanza lunga e noiosa, e dal punto di vista intuitivo non aggiunge niente di utile, quindi non verrà presentata.

2.5 Passo induttivo $E = RS$

Dati due automi

$$A_R = \langle Q_R, \Sigma_R, \delta_R, i_R, \{f_R\}\rangle \qquad A_S = \langle Q_S, \Sigma_S, \delta_S, i_S, \{f_S\}\rangle \qquad \text{con } Q_R \cap Q_S = \varnothing$$

che per ipotesi induttiva verificano (P1)–(P3) e riconoscono i linguaggi $L(A_R) = (L_R)$ e $L(A_S) = L(S)$, l'automa A_{RS} è descritto dal seguente diagramma di transizione:



Formalmente:

$$A_{RS} = \langle Q_R \cup Q_S, \Sigma_R \cup \Sigma_S, \delta, i_R, \{f_S\} \rangle$$

(per semplicità, viene qui omessa la definizione di δ).

Come sempre, la verifica di (P1)–(P3) è immediata; in particolare, il fatto che ci sia un solo stato finale è dato dalla costruzione dell'automa A_{RS} (che mette come finale solo f_S), mentre l'assenza di transizioni uscenti dallo stato finale e di transizioni entranti nello stato iniziale è garantita dall'ipotesi induttiva, secondo la quale tali proprietà valgono per A_R e A_S.

Per dimostrare invece che

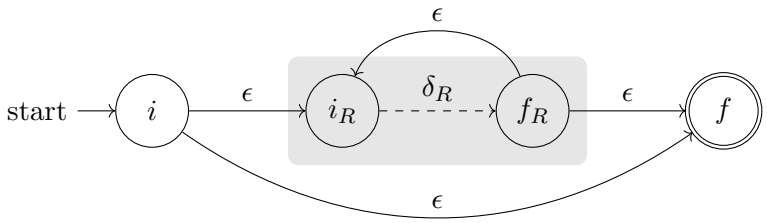
$$L(A_{RS}) = L(A_R) \cdot L(A_S) \overset{\text{(IH)}}{=} L(R) \cdot L(S) = L(RS)$$

si ragiona in modo simile a quanto fatto nel caso $E = R + S$, osservando che una stringa w è riconosciuta da A_{RS}, $w \in L(A_{RS})$, se e solo se è la concatenazione di due stringhe x e y, tali che $x \in L(A_R)$ e $y = L(A_S)$, ovvero $w = xy \in L(A_R) \cdot L(A_S)$:

- x determina un percorso di computazione da i_R a f_R;
- y determina un percorso di computazione da i_S a f_S;
- i due percorsi sono uniti da una ϵ-transizione, formando un percorso dallo stato iniziale a quello finale di A_{RS}.

2.6 Passo induttivo $E = R^*$

Dato un automa $A_R = \langle Q_R, \Sigma_R, \delta_R, i_R, \{f_R\} \rangle$, che per ipotesi induttiva verifica (P1)–(P3) e $L(A_R) = L(R)$, l'automa A_{R^*} è dato dal diagramma



ovvero è definito formalmente come:

$$A_{R^*} = \langle Q_R \cup \{i, f\}, \Sigma_R, \delta, i, \{f\} \rangle$$

(anche qui si omette la definizione di δ).

Le proprietà (P1)–(P3) sono immediatamente verificate per la costruzione dell'automa. Invece, bisogna dimostrare che A_{R^*} riconosca il linguaggio generato da R^*,

$$L(A_{R^*}) = L(A_R)^* \overset{\text{(IH)}}{=} L(R)^* = L(R^*)$$

e lo si fa trattando separatamente i due versi del seguente "se e solo se":

$$w \in L(A_{R^*}) \iff w \in L(A_R)^*$$

2.6.1 Verso $\implies$ della dimostrazione

Sia $w \in L(A_{R^*})$. Il fatto che questa stringa sia accettata implica l'esistenza di un percorso da i a f che consuma w:

$$i \overbrace{\to \cdots \to}^{w} f$$

Tale percorso potrebbe passare dagli stati i_R e f_R una o più volte, oppure zero volte, "saltando" invece direttamente da i a f, mediante l'ϵ-transizione $i \xrightarrow{\epsilon} f$. Sia k il numero di volte che il percorso passa per i_R. Si dimostra per induzione su k che $w \in L(A_R)^k \subseteq L(A_R)^*$.

- Se $k = 0$, il percorso seguito è $i \xrightarrow{\epsilon} f$, che non consuma input, quindi la stringa considerata, che per ipotesi viene accettata, deve essere $w = \epsilon$, e per definizione $\epsilon \in L(A_R)^0 \subseteq L(A_R)^*$.
- Se $k = h+1$, la stringa w può essere scomposta in $w = w^h w'$, dove w^h è riconosciuta passando h volte da i_R, da cui, per ipotesi induttiva, $w^h \in L(A_R)^h$. Una volta finito di consumare w^h, la computazione attraversa la ϵ-transizione $f_R \xrightarrow{\epsilon} i_R$ e si ferma all'$h+1$-esimo passaggio in i_R; a questo punto, sapendo che per ipotesi $w^h w'$ viene riconosciuta, si deduce che la computazione su w' deve portare da i_R a f_R,

  $$i \xrightarrow{\epsilon} i_R \overbrace{\to \cdots \to}^{w^h} f_R \xrightarrow{\epsilon} i_R \overbrace{\to \cdots \to}^{w'} f_R \xrightarrow{\epsilon} f$$

  il che implica $w' \in L(A_R)$. Allora, complessivamente:

  $$w = w^h w' \in L(A_R)^h \cdot L(A_R) = L(A_R)^{h+1} \subseteq L(A_R)^*$$

Intuitivamente, questa dimostrazione dice che, se la stringa w è stata riconosciuta "girando" un certo numero di volte (possibilmente zero) da i_R e f_R, allora è possibile scomporla in una serie di "pezzi" (zero pezzi, se $w = \epsilon$), ciascuno dei quali è riconosciuto da A_R. La loro concatenazione w appartiene dunque a una qualche potenza di $L(A_R)$, ovvero $w \in L(A_R)^*$.

2.6.2 Verso $\Longleftarrow$ della dimostrazione

Sia $w \in L(A_R)^*$. Per definizione, deve essere che $w \in L(A_R)^h$ per un qualche $h \geq 0$, e allora la stringa può essere scomposta come $w = w_1 w_2 \dots w_h$, dove ogni w_i è riconosciuta da A_R:

$$\forall i = 1, 2, \dots, h \quad w_i \in L(A_R)$$

Perciò, esiste un percorso che consuma w e porta da i a f in A_{R^*},

$$i \xrightarrow{\epsilon} i_R \overbrace{\rightarrow \cdots \rightarrow}^{w_1} f_R \xrightarrow{\epsilon} i_R \overbrace{\rightarrow \cdots \rightarrow}^{w_2} f_R \xrightarrow{\epsilon} i_R \ \cdots \ f_R \xrightarrow{\epsilon} i_R \overbrace{\rightarrow \cdots \rightarrow}^{w_h} f_R \xrightarrow{\epsilon} f$$

e questo implica che $w \in L(A_{R^*})$.

2.7 Passo induttivo $E = (R)$

Il caso delle parentesi è banale: dato per ipotesi induttiva $A_R = \langle Q_R, \Sigma_R, \delta_R, i_R, \{f_R\} \rangle$, che verifica (P1)–(P3) e $L(A_R) = L(R)$, è sufficiente porre $A_{(R)} = A_R$, poiché, per definizione, le parentesi non cambiano il linguaggio generato da un'espressione regolare:

$$L(A_{(R)}) = L(A_R) \overset{\text{(IH)}}{=} L(R) = L((R))$$