Azzolini Riccardo 2020-10-12

Linguaggi regolari e introduzione agli NFA

1 Linguaggio regolare

Un linguaggio $L \subseteq \Sigma^*$ è **regolare** se esiste un DFA $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ (sullo stesso alfabeto Σ su cui è definito L) tale che $L(A) = L$.

I linguaggi regolari hanno un ruolo importantissimo in informatica, e in particolare sono alla base di molti strumenti pratici.

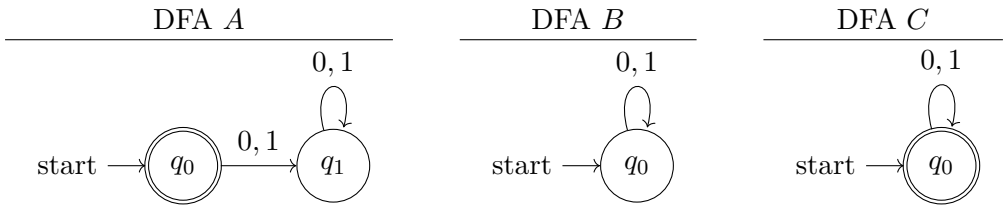
1.1 Definizioni alternative

Ci sono molti altri modi, tutti equivalenti, di definire i linguaggi regolari:

- altri dispositivi, oltre ai DFA, riconoscono i linguaggi regolari (ad esempio, NFA e ϵ-NFA);
- esistono formalismi che permettono di definire i linguaggi regolari mediante un processo di *generazione*, invece che di riconoscimento, delle stringhe di un linguaggio (ad esempio, le espressioni regolari e le grammatiche di tipo 3).

1.2 Esempi

Si considerino alcuni degli esempi di DFA mostrati in precedenza:



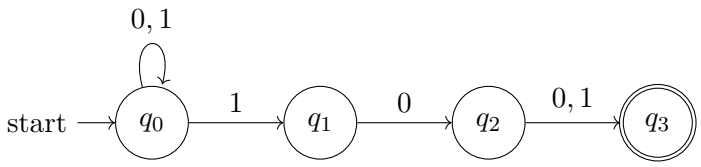
Si era dimostrato che

$$L(A) = \{\epsilon\} \qquad L(B) = \varnothing \qquad L(C) = \{0,1\}^*$$

dunque, per definizione, $L = \{\epsilon\}$, $L = \varnothing$ e $L = \{0,1\}^*$ sono linguaggi regolari.

2 Introduzione agli NFA

Si consideri il seguente diagramma di transizione:



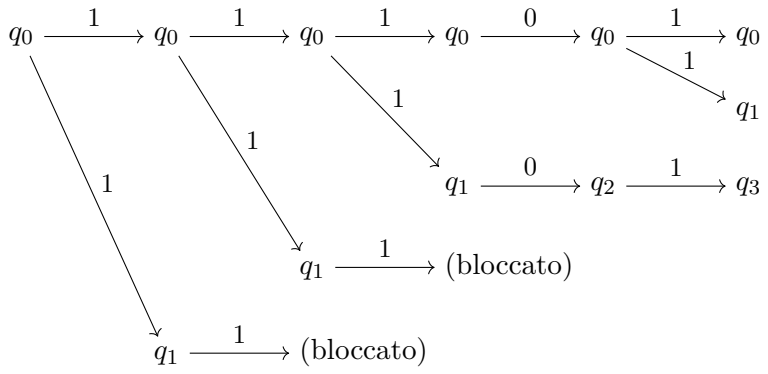
Esso non rispetta la definizione di DFA, perché:

- lo stato q_0 ha *due* transizioni uscenti etichettate 1, quindi non si può individuare un singolo valore da assegnare alla funzione $\delta(q_0, 1)$;
- lo stato q_1 non ha *nessuna* transizione uscente etichettata 1, ovvero nessun valore da assegnare a $\delta(q_1, 1)$;
- non ci sono transizioni uscenti da q_3 (quindi, ancora, non si riescono a determinare i valori della funzione di transizione).

Si supponga di voler ammettere questo tipo di dispositivo. Come si può dare a esso un senso, interpretando le situazioni "problematiche" appena descritte? La soluzione è sviluppare non una singola computazione, ma *tutte le possibili computazioni* dell'automa, stabilendo che:

- Quando uno stato ha più transizioni uscenti etichettate con lo stesso simbolo, ciò significa che ci sono più modi di proseguire la computazione. Ad esempio, se nello stato q_0 il simbolo di input è 1, la computazione può proseguire applicando la mossa $q_0 \xrightarrow{1} q_0$ oppure la mossa $q_0 \xrightarrow{1} q_1$.
- Quando invece uno stato non ha una transizione uscente etichettata con un determinato simbolo, se si ha tale simbolo in input la computazione risulta *"bloccata"* (non ha modo di procedere). Ad esempio, la computazione è bloccata se nello stato q_1 si ha il simbolo di input 1.

Sviluppare tutte le possibili computazioni dell'automa significa non ottenere più una sequenza lineare di mosse, ma piuttosto un **albero di computazione**. Ad esempio, le possibili computazioni sulla stringa $w = 11101$ sono rappresentate dall'albero



in cui:

- alcune computazioni sono bloccate (non hanno terminato la lettura della stringa);
- alcune computazioni sono arrivate in stati non finali (q_0 e q_1);
- una delle computazioni è invece arrivata in uno stato finale (q_3).

Siccome almeno una delle possibili computazioni ha portato a uno stato finale, la stringa $w = 11101$ è accettata dall'automa.

2.1 Non determinismo

Quello appena presentato è un **automa a stati finiti non deterministico** (**NFA**, *Nondeterministic Finite Automaton*). Come suggerisce il nome, questo tipo di automa modella una nozione di *non determinismo*, che matematicamente risulta molto naturale, ma è difficile da esprimere in modo intuitivo.

Una possibile interpretazione di un NFA è come un modo compatto per rappresentare insieme più DFA diversi: sviluppare l'albero di computazione dell'NFA su una stringa corrisponde allora a sviluppare le computazioni di tutti questi DFA sulla stringa, la quale viene accettata dall'NFA se e solo se è accettata da almeno uno dei DFA.

Dal punto di vista dell'implementazione, non si può definire un dispositivo fisico, concreto, che realizzi il comportamento di un NFA senza costruire l'intero albero di computazione. Costruire l'albero ha in generale un costo molto elevato, perché il numero di percorsi nell'albero è potenzialmente esponenziale (mentre la computazione di un DFA è di dimensione lineare: data una stringa in input w, esso fa esattamente $|w|$ passi). Il fatto di costruire l'intero albero non è però richiesto dalla definizione di NFA: si potrebbe immaginare un dispositivo in grado di costruire solo il percorso accettante (quando esso esiste), che in quanto singolo percorso ha una lunghezza lineare.

Se da un lato gli NFA sono più difficili da implementare, dall'altro è molto più facile, dato un linguaggio regolare, costruire un NFA che lo riconosca, piuttosto che costruire direttamente un DFA. Fortunatamente, si dimostrerà che gli NFA sono *equivalenti* ai

DFA, (NFA $\approx$ DFA): esiste un NFA che riconosce un linguaggio L se e solo se esiste un DFA che riconosce L, e dall'uno si può costruire l'altro mediante un apposito algoritmo. Così, dato un linguaggio regolare, il modo più semplice per implementarne il riconoscimento sarà prima costruire un NFA (in particolare, un ϵ-NFA, un altro tipo di automa a stati finiti che è a sua volta equivalente a un NFA), e poi convertirlo in un DFA.

Tuttavia, si vedrà anche che l'equivalenza vale solo dal punto di vista espressivo, e non invece da quello delle risorse richieste: un DFA corrispondente a un determinato NFA può avere, nel caso peggiore, un numero di stati esponenziale rispetto a quello dell'NFA. Questo è proprio il "costo" del simulare deterministicamente il non determinismo: come detto prima, non si può costruire un dispositivo concreto che realizzi il non determinismo in modo efficiente.

2.2 Funzione di transizione

La funzione di transizione di un DFA ha la forma $\delta : Q \times \Sigma \to Q$, cioè associa a ogni coppia stato-simbolo lo stato in cui evolve il DFA. Per formalizzare un NFA, questa funzione può essere estesa associando a ciascuna coppia stato-simbolo un *insieme* di stati,

$$\delta : Q \times \Sigma \to 2^Q$$

(dove 2^Q, indicato anche con $\mathscr{P}(Q)$, è l'insieme delle parti di Q, cioè l'insieme di tutti i sottoinsiemi di Q).

- Quando uno stato q non ha transizioni uscenti etichettate con il simbolo a, si pone $\delta(q, a) = \varnothing$. Ad esempio, nell'NFA mostrato prima, si ha $\delta(q_1, 1) = \varnothing$.
- Quando q ha più transizioni uscenti etichettate con a, che conducono agli stati $q', q'', \ldots$, si pone $\delta(q, a) = \{q', q'', \ldots\}$. Nell'esempio di prima, $\delta(q_0, 1) = \{q_0, q_1\}$.
- Se q ha una sola transizione uscente etichettata con a (come in un DFA), che conduce a uno stato q', si pone $\delta(q, a) = \{q'\}$ (dove $\{q'\}$ è un insieme *singoletto*, cioè con un singolo elemento). Per esempio, nell'NFA di prima, si ha $\delta(q_0, 0) = \{q_0\}$.