Provare che un linguaggio sia regolare

L'idea è la seguente:
- avere uno stato iniziale composto da una tripla (composta da:

1) q_0 (stato iniziale e/o stato attuale)
2) simbolo che uso per raggiungere lo stato fuori dalle parentesi
3) eventuale flag che indica l'input dell'automa processato in quel momento nel modo scritto
4) simbolo raggiunto dopo una transizione

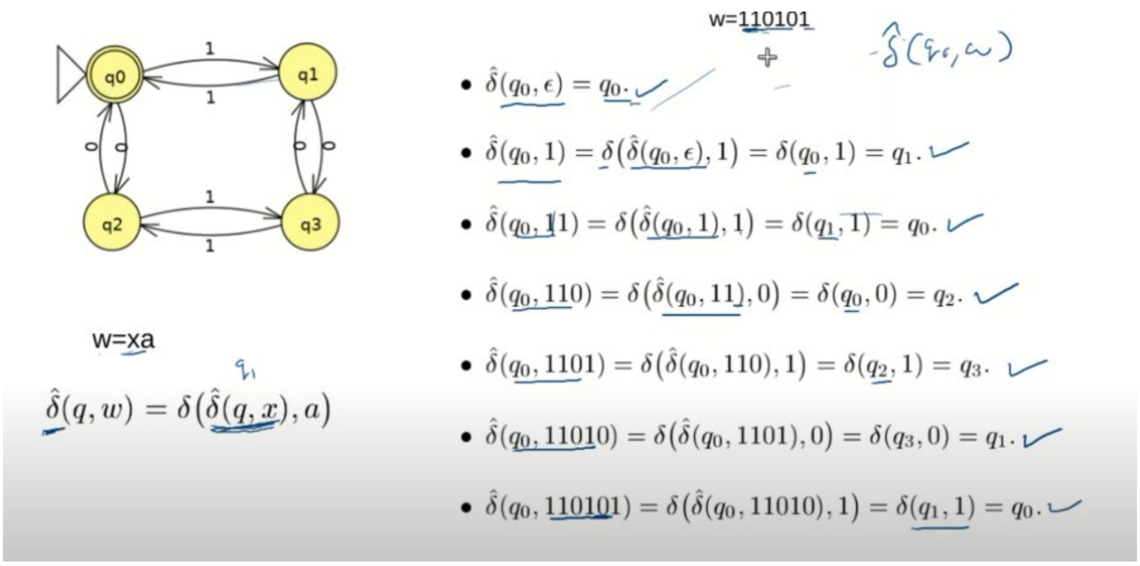
Nell'esempio di Giulio:
$\delta((x,y,A),a) = (\delta_A(x,a), y, B)$

Detto a parole: (leggere testualmente)
Da "x" vado ad "y" dell'automa A *usando come simbolo "a"* arrivando verso "y" dell'automa B.

Se ho un insieme di stati (esempio and logico), allora li metto tutti insieme, quindi:
$\delta((r_L, r_M), 0)$ (che significa, tutti gli stati dei linguaggi R ed M che coinvolgono 1)
$\delta((r_L, r_M), 1)$ (che significa, tutti gli stati dei linguaggi R ed M che coinvolgono 1)

- avere un insieme di stati finali (tutte le coppie raggiungibili quindi)

$F = F_A \times F_B$



Dal link https://cs.stackexchange.com/questions/1331/how-to-prove-a-language-is-regular

Another method, not covered by the answers above, is *finite automaton transformation*. As a simple example, let us show that the regular languages are closed under the *shuffle* operation, defined as follows:

$$L_1 \, S \, L_2 = \{x_1 y_1 \ldots x_n y_n \in \Sigma^* : x_1 \ldots x_n \in L_1, y_1 \ldots y_n \in L_2\}$$

You can show closure under shuffle using closure properties, but you can also show it directly using DFAs. Suppose that $A_i = \langle \Sigma, Q_i, F_i, \delta_i, q_{0i} \rangle$ is a DFA that accepts L_i (for $i = 1, 2$). We construct a new DFA $\langle \Sigma, Q, F, \delta, q_0 \rangle$ as follows:

- The set of states is $Q_1 \times Q_2 \times \{1, 2\}$, where the third component remembers whether the next symbol is an x_i (when 1) or a y_i (when 2).
- The initial state is $q_0 = \langle q_{01}, q_{02}, 1 \rangle$.
- The accepting states are $F = F_1 \times F_2 \times \{1\}$.
- The transition function is defined by $\delta(\langle q_1, q_2, 1 \rangle, \sigma) = \langle \delta_1(q_1, \sigma), q_2, 2 \rangle$ and $\delta(\langle q_1, q_2, 2 \rangle, \sigma) = \langle q_1, \delta_2(q_2, \sigma), 1 \rangle$.

Tradotto a parole:

- ho un insieme di stati definiti come transizioni possibili “1” e “2”
- lo stato iniziale è dato dall’unione 0/1 oppure 0/2 (essendo shuffle, rimescola; *basti leggere*)
- stati accettanti; il simbolo di input viene almeno una volta ripetuto (finisco con 1)
- funzione di transizione <u>scritta letteralmente</u>
  Parto da q1 con il simbolo “1” , arrivo verso q2 con il simbolo 2 passando per σ

Guessing often involves also verifying. One simple example is closure under *rotation*:

$$R(L) = \{yx \in \Sigma^* : xy \in L\}.$$

Suppose that L is accepted by the DFA $\langle \Sigma, Q, F, \delta, q_0 \rangle$. We construct an NFA $\langle \Sigma, Q', F', \delta', q_0' \rangle$, which operates as follows. The NFA first guesses $q = \delta(q_0, x)$. It then verifies that $\delta(q, y) \in F$ and that $\delta(q_0, x) = q$, moving from y to x non-deterministically. This can be formalized as follows:

- The states are $Q' = \{q_0'\} \cup Q \times Q \times \{1, 2\}$. Apart from the initial state q_0', the states are $\langle q, q_{curr}, s \rangle$, where q is the state that we guessed, q_{curr} is the current state, and s specifies whether we are at the y part of the input (when 1) or at the x part of the input (when 2).
- The final states are $F' = \{\langle q, q, 2 \rangle : q \in Q\}$: we accept when $\delta(q_0, x) = q$.
- The transitions $\delta'(q_0', \epsilon) = \{\langle q, q, 1 \rangle : q \in Q\}$ implement guessing q.
- The transitions $\delta'(\langle q, q_{curr}, s \rangle, \sigma) = \langle q, \delta(q_{curr}, \sigma), s \rangle$ (for every $q, q_{curr} \in Q$ and $s \in \{1, 2\}$) simulate the original DFA.
- The transitions $\delta'(\langle q, q_f, 1 \rangle, \epsilon) = \langle q, q_0, 2 \rangle$, for every $q \in Q$ and $q_f \in F$, implement moving from the y part to the x part. This is only allowed if we've reached a final state on the y part.

Tradotto a parole:

- ho un insieme di stati definiti come transizioni possibili “1” e “2”
- essendo la rotazione, possiamo avere sia simbolo 1 che 2 come iniziale
- lo stato finale è 2 se inizio con 1, similmente sarà 1 se inizio con 2
- tradotta a parole la transizione:
  inizio con “q”, arrivo verso “q0” con il simbolo 2 (se parto con 1)
  inizio con “q”, arrivo verso “q0” con il simbolo 1 (se parto con 2)

Altrimenti si può dimostrare con un ε-NFA:

4. Sia L un linguaggio regolare su un alfabeto Σ e dimostrate che il seguente linguaggio è regolare:

$$\mathit{suffixes}(L) = \{y \mid xy \in L \text{ per qualche stringa } x \in \Sigma^*\}$$

Intuitivamente, $\mathit{suffixes}(L)$ è il linguaggio di tutti i suffissi delle parole che stanno in L.

> Per dimostrare che $\mathit{suffixes}(L)$ è regolare basta mostrare come costruire un automa a stati finiti che riconosce $\mathit{suffixes}(L)$ a partire dall'automa a stati finiti che riconosce L.
>
> Sia quindi $A = (Q, \Sigma, q_0, \delta, F)$ un automa a stati finiti che riconosce il linguaggio L. Costruiamo un ε-NFA $B = (Q \cup \{q_0'\}, \Sigma, q_0', \delta_B, F)$ che ha uno stato in più di A. Chiamiamo q_0' questo nuovo stato e gli assegnamo il ruolo di stato iniziale di B. La funzione di transizione del nuovo automa aggiunge una ε-transizione dal nuovo stato iniziale q_0' verso ogni stato di Q che è raggiungibile dal vecchio stato iniziale. Il resto delle transizioni rimane invariato. Gli stati finali sono gli stessi di A.

Esempio completo

Ex 5.2

Siano L e M due linguaggi regolari su alfabeto {0,1}. Dimostrare che il linguaggio L&M $= \{x\&y | x \in L, y \in M, |x| = |y|\}$, dove x&y e' l'and logic bit a bit. Per esempio, 101&001 = 001.

I linguaggi sono regolari, quindi abbiamo a disposizione due automi:

- $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$
- $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$

Possiamo dedurre il seguente comportamento:

- Quando l'input e' 1 sappiamo che gli automi stanno processando entrambi il valore 1.
- Se il simbolo e' 0, non sappiamo di preciso quale sia il valore processato - infatti a 0 corrispondono tre coppie di valori.

Definiamo x e y come gli stati attuali di D_A e D_B e definiamo la funzione di transizione come segue.

- $\delta((x,y),1) = (\delta(x,1), \delta(y,1))$
- $\delta((x,y),0) = ((\delta(x,1), \delta(y,0)), (\delta(x,0), \delta(y,1)), (\delta(x,0), \delta(y,0))$

Se D_A automa accetta la parola, esistera' un percorso che porta ad uno stato accettante (stessa cosa per D_B).

Per completare l'automa, dobbiamo definirne le altre componenti:

- $\Sigma = \{0, 1\}$
- $Q_S = Q_A$ x Q_B
- q = (q_A, q_B)
- F = F_A x F_B