DFA - Slide

Gabriel Rovesti

1 Consegna 1

Linguaggio: L'insieme di tutte e sole le stringhe che contengono un numero pari di zeri e un numero pari di uni (alfabeto $\{0, 1\}$).

Soluzione

Per tenere traccia in maniera deterministica della parità di zeri e di uni, è sufficiente "ricordare" (con gli stati) se finora abbiamo letto un numero pari o dispari di `0` e un numero pari o dispari di `1`. Definiamo allora 4 stati:

$$Q = \{\, (p0, p1),\ (p0, d1),\ (d0, p1),\ (d0, d1)\}$$

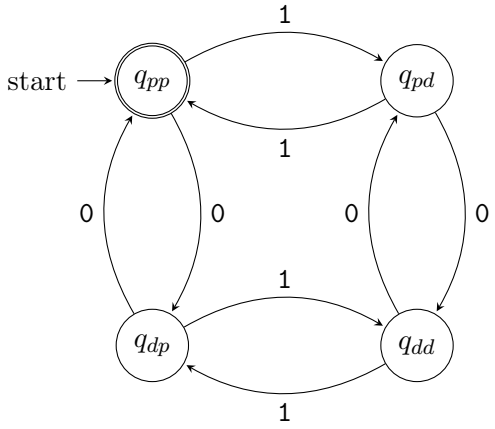
dove `p0` significa "letti finora zero in numero pari", `d0` "letti zero in numero dispari" (analogamente per `p1`/`d1` riguardo ai `1`). Lo stato iniziale è $(p0, p1)$, in cui non abbiamo ancora letto nulla (numero di `0` e di `1` $= 0$, entrambi pari); è anche finale, perché ci servono entrambi i conteggi pari. Le transizioni si definiscono così:

- Leggere `0` inverte la parità dei `0` ma non tocca la parità dei `1`. - Leggere `1` inverte la parità dei `1` ma non tocca quella dei `0`.

Graficamente, etichetteremo i 4 stati così:

$$q_{pp} = (p0, p1), \quad q_{pd} = (p0, d1), \quad q_{dp} = (d0, p1), \quad q_{dd} = (d0, d1).$$

La funzione di transizione δ può essere illustrata in un diagramma come segue:



Si vede che q_{pp} è l'unico stato *accepting* (poiché vogliamo parità di zero e uno). Questo DFA accetta esattamente le stringhe con numero pari di `0` e numero pari di `1`.

2 Consegna 2

Linguaggio: L'insieme di tutte le stringhe (su {0,1}) che terminano con `00`.

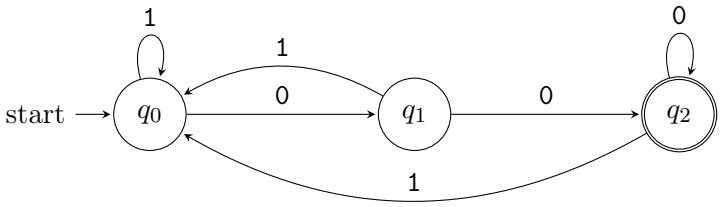
Soluzione

Qui basta "tenere d'occhio" gli ultimi due simboli letti. Progettiamo un piccolo DFA con 3 stati significativi:

- q_0: stato iniziale, significa "non ho ancora riconosciuto di terminare con 0 (oppure la stringa è vuota oppure termina con 1)".
- q_1: so che l'ultimo simbolo letto è `0` ma non ho ancora `00`.
- q_2: so che gli ultimi due simboli letti sono `00` (stato *finale*).

Le transizioni: - Da q_0, se leggo `0`, passo a q_1; se leggo `1` , rimango in q_0 (continuo a non finire in `0`). - Da q_1, se leggo `0`, allora ho "finisco in `00`", vado a q_2; se leggo `1` , torno a q_0. - Da q_2 (ultimo due simboli `00`), se leggo `0` resto in q_2 (continuo a finire in `00`), se leggo `1` torno a q_0 (ora finisco in `01`).

Diagramma:



Stato finale è solo q_2, perché significa "la stringa letta fin qui finisce con `00`".

3 Consegna 3

Linguaggio: L'insieme di tutte le stringhe che contengono *esattamente* tre zeri (anche non consecutivi).

Soluzione

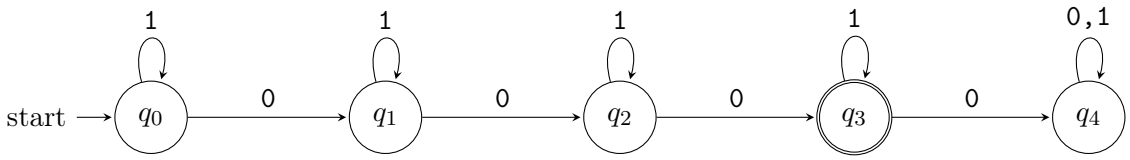
Costruiamo un automa che conti quanti `0` ha visto. Se il totale supera 3, andiamo in uno stato di *trap*. Formalmente:

- q_0: "letti 0 zeri". (Iniziale)
- q_1: "letti 1 zero".
- q_2: "letti 2 zeri".
- q_3: "letti 3 zeri". (questo è finale)
- q_4: "letti più di 3 zeri" (trap state, da cui non si esce più).

Le transizioni:

- Da q_0 leggendo `0` $\to q_1$, leggendo `1` $\to q_0$.
- Da q_1 leggendo `0` $\to q_2$, leggendo `1` $\to q_1$.
- Da q_2 leggendo `0` $\to q_3$, leggendo `1` $\to q_2$.
- Da q_3 leggendo `0` $\to q_4$, leggendo `1` $\to q_3$.
- q_4 è trap: se leggo `0` o `1`, rimango in q_4.

q_3 è lo *stato finale* unico, perché significa "è stato letto esattamente un totale di 3 zeri". Diagramma:



4 Consegna 4

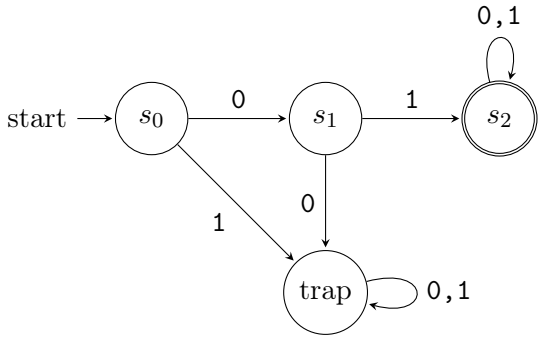
Linguaggio: L'insieme di tutte le stringhe che *cominciano* oppure *finiscono* (o entrambe) con `01`.

Soluzione (cenno con 2 DFA e unione)

L'insieme richiesto è

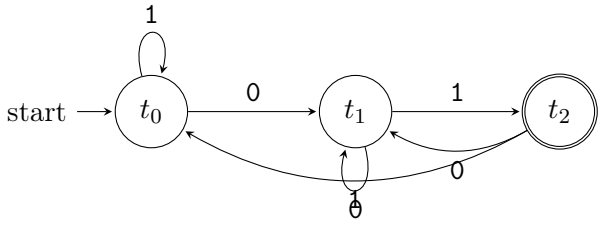
$$L = \{\text{stringhe che iniziano con } 01\} \cup \{\text{stringhe che terminano con } 01\}.$$

Ciascuno dei due linguaggi è *regolare* e anzi definibile da un piccolo DFA. L'automa che accetta "inizia con `01`" (denotiamolo M_{start}) può essere ad esempio:



perché se la stringa non inizia con '0' o non prosegue con '1', finiamo in uno stato *trap*. Se invece leggiamo "`01`" come primi due simboli, passiamo nello stato finale s_2 e continuiamo ad accettare qualsiasi coda (restando in s_2).

Analogamente, il DFA per "termina con `01`" (chiamiamolo M_{end}) è:



dove t_2 rappresenta lo stato "gli ultimi due simboli letti sono `01`". Notare che t_0 non è accettante, perché la stringa non termina (ancora) con `01`; t_1 neppure.

L'automa complessivo $M_{\text{start}} \cup M_{\text{end}}$ (unione di due linguaggi) in teoria si può costruire prendendo il *prodotto* dei due DFA e marcando come finali tutte le coppie (s, t) in cui s è finale in M_{start} *o* t è finale in M_{end}. In

pratica, si può disegnare anche un NFA più piccolo tramite ε-transizioni (il che a volte rende più immediata la comprensione). In ogni caso, *esiste* un DFA risolutivo (il suo *minimale* ha effettivamente 8 stati), ma talvolta, per mero esercizio, è sufficiente uno schema come sopra: due automi distinti e l'osservazione che il linguaggio cercato è la loro unione.