Programma sintetico

- Nozioni preliminari
- **Automi Finiti**
- Macchine di Turing
- Limiti delle macchine di Turing
- La tesi di Church-Turing
- Le classi P e NP

Che cos'è un computer?

Modelli di computazione (computer ideali che permettono di studiare alcuni aspetti del computer reale: quali problemi possiamo risolvere con un computer, quali non possiamo risolvere?)

Automi finiti

Un automa finito è un modello di computer con una quantità estremamente limitata di memoria.

Che cosa possiamo fare con una macchina del genere? Tante cose utili.

dispositivi elettromeccanici: lavastoviglie, sistemi di controllo, lettori automatici, erogatori di bibite e snack, ...

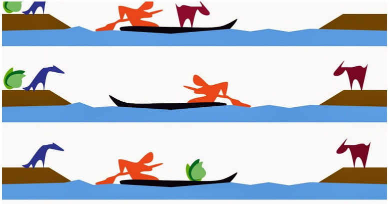
software: parsing (analisi sintattica) di compilatori, ricerca di parole in un file, ...

Un problema classico di logica

- un uomo vuole trasportare sull'altra riva di un fiume una pecora, un lupo e un cavolo
- ha una barca che può contenere solo l'uomo più uno dei suoi possedimenti
- il lupo mangia la pecora se lasciati soli insieme
- la pecora mangia il cavolo se lasciati soli insieme
- come può attraversare fiume senza perdite?

Un problema classico di logica

Le mosse possono essere rappresentate da simboli di un alfabeto $\Sigma = \{l, p, c, n\}$:

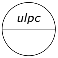


- l: l'uomo attraversa con il **lupo**
- p: l'uomo attraversa con la **pecora**
- c: l'uomo attraversa con il **cavolo**
- n: l'uomo attraversa con **niente**

Una soluzione è data dalla stringa `pncplnp` $\Longrightarrow$ attraversa con il lupo, torna con niente, attraversa con il cavolo, ...

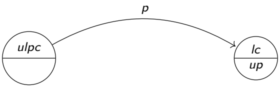
Un problema classico di logica

Ogni mossa porta da una configurazione (**stato**) all'altro: provoca una **transizione** da uno stato q a uno stato q'.
La situazione iniziale vede tutti sulla stessa riva del fiume:



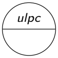
Se l'uomo traghetta la pecora sull'altra sponda, la situazione diventerà:

Se da quest'ultima situazione l'uomo riporta indietro la pecora sulla riva di partenza, la situazione ritornerà quella iniziale.
Possiamo rappresentare queste transizioni provocate dalla mossa p nel seguente diagramma:



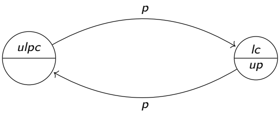
Un problema classico di logica

Ogni mossa porta da una configurazione (**stato**) all'altro: provoca una **transizione** da uno stato q a uno stato q'.
La situazione iniziale vede tutti sulla stessa riva del fiume:



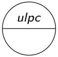
Se l'uomo traghetta la pecora sull'altra sponda, la situazione diventerà:

Se da quest'ultima situazione l'uomo riporta indietro la pecora sulla riva di partenza, la situazione ritornerà quella iniziale.
Possiamo rappresentare queste transizioni provocate dalla mossa p nel seguente diagramma:

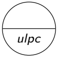


Un problema classico di logica

Una soluzione al problema può essere rappresentata da un diagramma di transizioni che parta dalla situazione iniziale:

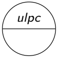


e, evitando tutte le situazioni nelle quali il lupo mangia la pecora o la capra mangia il cavolo, termini nella configurazione finale:

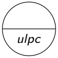


Un problema classico di logica

Una soluzione al problema può essere rappresentata da un diagramma di transizioni che parta dalla situazione iniziale:



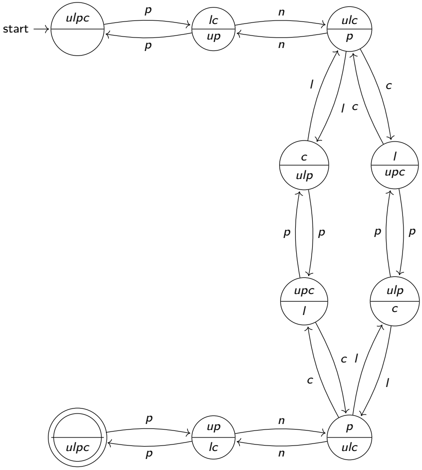
e, evitando tutte le situazioni nelle quali il lupo mangia la pecora o la capra mangia il cavolo, termini nella configurazione finale:



Proviamo a rappresentare le soluzioni date dalle seguenti stringhe:

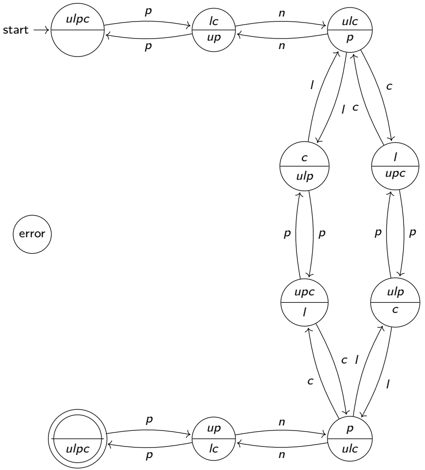
- `pnlpcnp`
- `pncplnp`

Un problema classico di logica

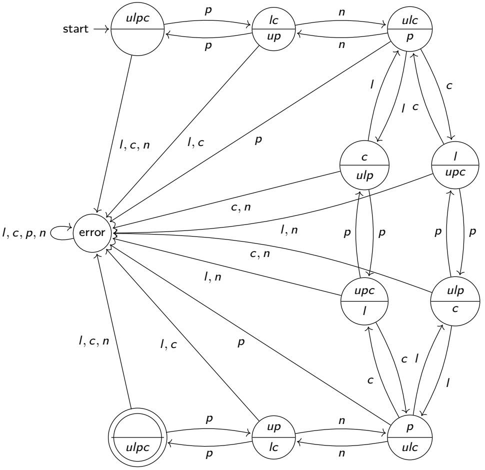


Un problema classico di logica

Possiamo rappresentare anche le soluzioni errate (facendole terminare in uno stato *error*)?



Nota bene: ora ogni stato ha un arco per ogni possibile mossa!



Un problema classico di logica

- ogni stringa $x \in \{l, p, c, n\}^*$ corrisponde a un cammino $p(x)$ sul diagramma e viceversa
- una stringa $x \in \{l, p, c, n\}^*$ è una soluzione se e solo se il cammino corrispondente $p(x)$ parte dallo stato iniziale e termina nello stato finale del diagramma
- Insieme delle soluzioni:

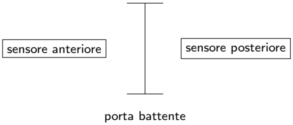
$$\{x \in \{l, p, c, n\}^* \mid p(x) \text{ termina nello stato finale del diagramma}\}$$

Modello semplice di calcolatore avente una quantità finita di memoria. È noto anche come **macchina a stati finiti**.

Idea di base del funzionamento.
Sia $\Sigma = \{a, b, c, d\}$ un insieme finito di simboli, detto **alfabeto**.

- Input = stringa w sull'alfabeto Σ.
- Legge i simboli di w da sinistra a destra.
- Dopo aver letto l'ultimo simbolo, l'automa indica se accetta o rifiuta la stringa input w.

Esempio: sistema di controllo porta automatica

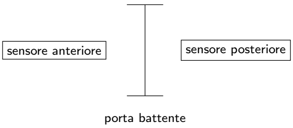


Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Esempio: sistema di controllo porta automatica



Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

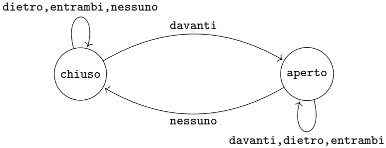
Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Regola:

- se la porta è chiusa, si apre solo se arriva una persona davanti;
- se la porta è aperta, si chiude solo se non c'è nessuno.

Esempio: sistema di controllo porta automatica



Il sistema di controllo può trovarsi in uno di due possibili **stati**: `aperto` o `chiuso`.

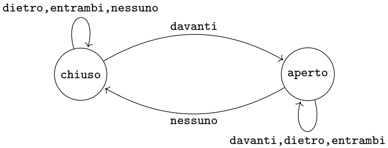
Le situazioni di input rilevate dai sensori sono le seguenti.

- `davanti`: persona davanti la soglia
- `dietro`: persona dietro la soglia
- `entrambi`: persona davanti e persona dietro la soglia
- `nessuno`: nessuna persona in prossimità della porta (né davanti né dietro)

Regola:

- se la porta è chiusa, si apre solo se arriva una persona davanti;
- se la porta è aperta, si chiude solo se non c'è nessuno.

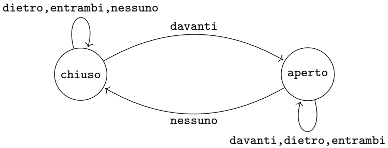
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input:

Allora passerà attraverso la sequenza di stati: `chiuso`,

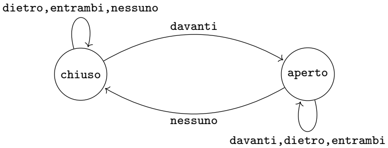
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`,

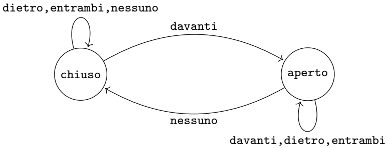
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`,

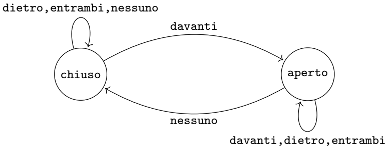
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`, `chiuso`,

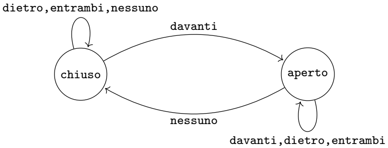
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`, `chiuso`, `aperto`,

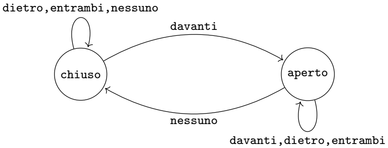
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`, `chiuso`, `aperto`, `aperto`,

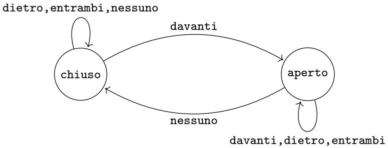
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`, `chiuso`, `aperto`, `aperto`, `chiuso`,

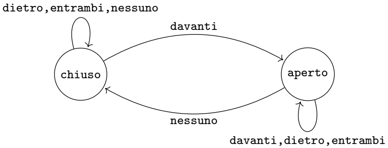
Esempio: sistema di controllo porta automatica



Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`,

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`, `chiuso`, `aperto`, `aperto`, `chiuso`, `chiuso`,

Esempio: sistema di controllo porta automatica

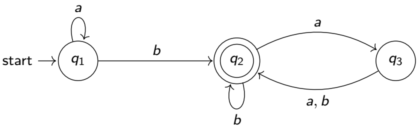


Supponiamo che il sistema parta nella situazione (stato) di `chiuso` e riceva la seguente sequenza di input: `davanti`, `dietro`, `nessuno`, `davanti`, `entrambi`, `nessuno`, `dietro`, `nessuno`.

Allora passerà attraverso la sequenza di stati: `chiuso`, `aperto`, `aperto`, `chiuso`, `aperto`, `aperto`, `chiuso`, `chiuso`, `chiuso`.

Automa finito deterministico (DFA)

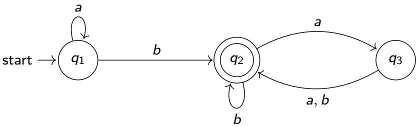
Diagramma di stato dell'automa M_1



- alfabeto $\Sigma = \{a, b\}$
- insieme degli stati $Q = \{q_1, q_2, q_3\}$
- stato iniziale q_1
- stato finale q_2
- per ogni stato, se si legge un simbolo dell'alfabeto (a o b), il diagramma specifica in quale stato si transisce

Automa finito deterministico (DFA)

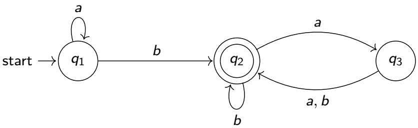
Diagramma di stato dell'automa M_1



Se arriva in input una stringa w, il DFA si comporta come segue:

- partendo dallo stato iniziale q_1, e leggendo un simbolo alla volta, il DFA passa da uno stato all'altro
- se lo stato in cui approda leggendo l'ultimo simbolo è finale, cioè q_2, allora la stringa è accettata, altrimenti è rifiutata

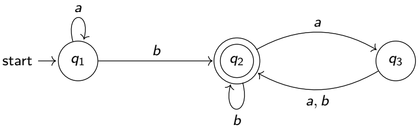
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1

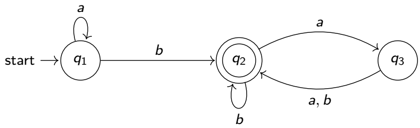
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1
- legge `a` e resta in q_1

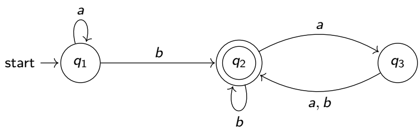
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1
- legge `a` e resta in q_1
- legge `b` e transisce da q_1 a q_2

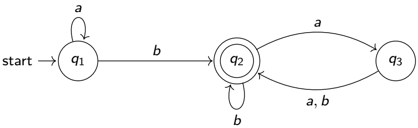
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1
- legge `a` e resta in q_1
- legge `b` e transisce da q_1 a q_2
- legge `b` e resta in q_2

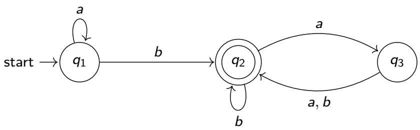
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1
- legge `a` e resta in q_1
- legge `b` e transisce da q_1 a q_2
- legge `b` e resta in q_2
- legge `a` e transisce da q_2 a q_3

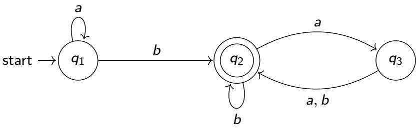
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1
- legge `a` e resta in q_1
- legge `b` e transisce da q_1 a q_2
- legge `b` e resta in q_2
- legge `a` e transisce da q_2 a q_3
- legge `a` e transisce da q_3 a q_2

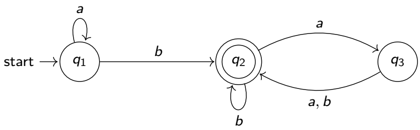
Automa finito deterministico (DFA)



stringa input: `abbaa`

- inizia nello stato q_1
- legge `a` e resta in q_1
- legge `b` e transisce da q_1 a q_2
- legge `b` e resta in q_2
- legge `a` e transisce da q_2 a q_3
- legge `a` e transisce da q_3 a q_2
- la stringa è accettata perché alla fine dell'input l'automa si trova in q_2 che è uno stato accettante

Automa finito deterministico (DFA)

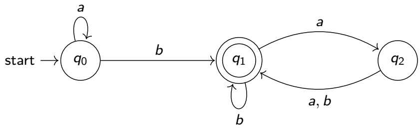


abbaa è accettata (dopo aver letto l'ultimo simbolo il DFA si trova nello stato accettante: q_2)

abaaa è rifiutata (dopo aver letto l'ultimo simbolo il DFA si trova nello stato q_3 che non è accettante)

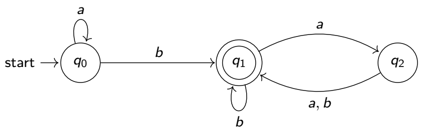
ϵ è rifiutata (lo stato iniziale q_1 non è accettante)

Automa finito deterministico: definizione formale



Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$

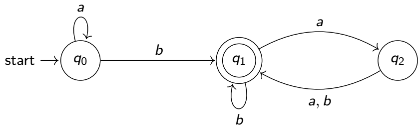
Automa finito deterministico: definizione formale



Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$

- Q è un insieme finito chiamato l'**insieme degli stati**

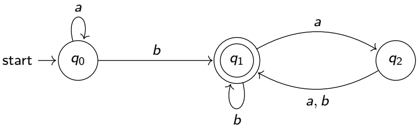
Automa finito deterministico: definizione formale



Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**

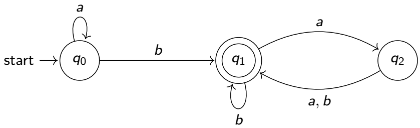
Automa finito deterministico: definizione formale



Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma \to Q$ è la **funzione di transizione**

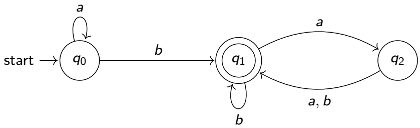
Automa finito deterministico: definizione formale



Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma \to Q$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**

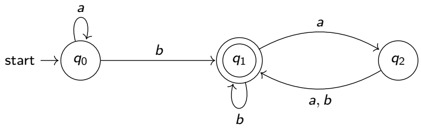
Automa finito deterministico: definizione formale



Un automa finito è una quintupla $(Q, \Sigma, \delta, q_0, F)$

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma \to Q$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'**insieme degli stati accettanti** (o **finali**)

Automa finito deterministico: definizione formale



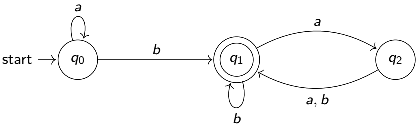
La funzione di transizione $\delta : Q \times \Sigma \rightarrow Q$ specifica per ogni stato e per ogni simbolo input, in quale stato si transisce.

$\delta(q_i, a) \in Q$ è lo stato in cui si troverà il DFA quando, trovandosi nello stato q_i, legge il simbolo a. Ad esempio, nell'automa rappresentato sopra $\delta(q_1, a) = q_2$.

Perché si dice **deterministico**?

- Per ogni stato esiste una ed una sola transizione per ciascun simbolo dell'alfabeto

Automa finito deterministico: definizione formale



Definiamo $M = (Q, \Sigma, \delta, q_0, F)$.

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- δ è così descritta:

	a	b
$\rightarrow q_0$	q_0	q_1
$* \, q_1$	q_2	q_1
q_2	q_1	q_1

- q_0 è lo stato iniziale
- $F = \{q_1\}$

Come computa un DFA?

Sia $M_1 = (Q, \Sigma, \delta, q_0, F)$ un DFA. Consideriamo la stringa input $w = w_1 w_2 \cdots w_n$, dove $w_i \in \Sigma$ per $i = 1, 2, \ldots, n$.

M accetta la stringa w se esiste una sequenza di stati $r_0, r_1, \ldots, r_n$ tale che:

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ per $i = 0, \ldots, n-1$
- $r_n \in F$

Se A è l'insieme di tutte le stringhe che la macchina accetta, diciamo che A è il **linguaggio della macchina** M e scriviamo $L(M) = A$.
Diciamo che M **riconosce** A.

Linguaggio della macchina

Definizione

Se A è l'insieme di tutte le stringhe che la macchina M accetta, diciamo che A è il **linguaggio della macchina** *M e scriviamo*

$$L(M) = A.$$

Diciamo anche che M **riconosce** (o **accetta**) A.

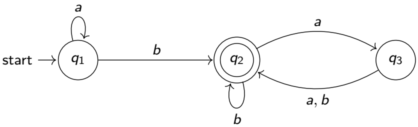
Ogni macchina riconosce sempre e soltanto un linguaggio. Se la macchina non accetta alcuna stringa, riconosce ancora un linguaggio: il linguaggio vuoto. In questo caso

$$L(M) = \emptyset.$$

Linguaggio regolare

Definizione

Un linguaggio è chiamato **linguaggio regolare** *se esiste un automa finito che lo riconosce.*



Qual è il linguaggio riconosciuto da questo automa M_1?

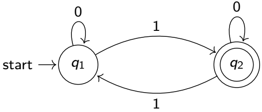
$L(M_1)$ è l'insieme di tutte le stringhe della forma

$$\{a\}^* b \; \{b, ab, aa\}^*.$$

Equivalentemente
$L(M_1) = \{w |\; w$ contiene almeno un b e un numero pari di a segue l'ultimo $b\}$
(ricordare che 0 è pari).

Esempio 1

Quale linguaggio riconosce il seguente DFA M_1?



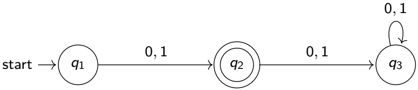
Esempi di stringhe accettate: 1, 111, 00010000, 001001001000.
Esempi di stringhe rifiutate: 0, 11, 010010, 0010010010100.

$L(M_1) = \{w \in \Sigma^* | \ w \text{ contiene un numero dispari di } 1\}$

Esercizio: dare un DFA che riconosca il linguaggio su $\Sigma = \{0, 1\}$ formato da tutte le stringhe con un numero pari di 1.

Esempio 2

Quale linguaggio riconosce il seguente DFA M_2 sull'alfabeto $\Sigma = \{0, 1\}$?



Esempi di stringhe accettate: 0, 1.
Esempi di stringhe rifiutate: 00, 01, 000, 101.

$L(M_2) = \{w \in \Sigma^* |\ |w| = 1\}$

Il complemento di $L(M_2)$ è il linguaggio di tutte le stringhe su Σ che hanno lunghezza diversa da 1. Cioè $C(L(M_2)) = \{w \in \Sigma^* |\ |w| \neq 1\}$.

Esempio 3

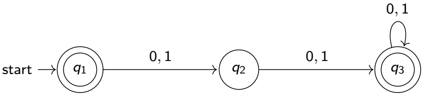
Progettiamo un DFA M_3 che riconosca

$$C(L(M_2)) = \{w \in \Sigma^* |\ |w| \neq 1\} = \{\epsilon\} \cup \{w \in \Sigma^* |\ |w| > 1\}.$$

Esempio 3

Progettiamo un DFA M_3 che riconosca

$$C(L(M_2)) = \{w \in \Sigma^* |\ |w| \neq 1\} = \{\epsilon\} \cup \{w \in \Sigma^* |\ |w| > 1\}.$$

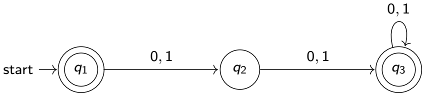


- M_3 ha più di uno stato accettante
- lo stato iniziale di M_3 è anche stato accettante

Esempio 3

Progettiamo un DFA M_3 che riconosca

$$C(L(M_2)) = \{w \in \Sigma^* \mid |w| \neq 1\} = \{\epsilon\} \cup \{w \in \Sigma^* \mid |w| > 1\}.$$



- M_3 ha più di uno stato accettante
- lo stato iniziale di M_3 è anche stato accettante

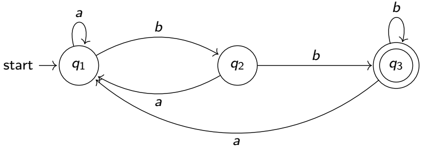
Osservazione

Un DFA accetta ϵ se e solo se il suo stato iniziale è accettante.

Esempio 4

Sia M_4 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

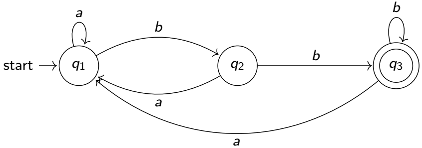
Quale linguaggio riconosce M_4?



Esempio 4

Sia M_4 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_4?



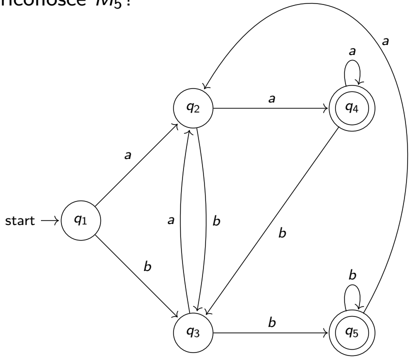
M_4 accetta tutte (e sole) le stringhe su Σ che terminano con bb.

$$L(M_4) = \{w \mid sbb \text{ con } s \in \Sigma^*\}.$$

Esempio 5

Sia M_5 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

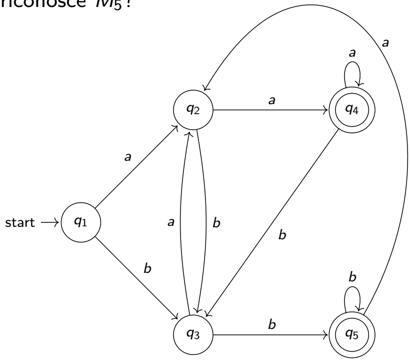
Quale linguaggio riconosce M_5?



Esempio 5

Sia M_5 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_5?

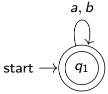


$$L(M_5) = \{w |\ w = saa \text{ oppure } w = sbb \text{ per qualche stringa } s\}$$

Esempio 6

Sia M_6 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

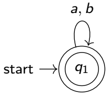
Quale linguaggio riconosce M_6?



Esempio 6

Sia M_6 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_6?



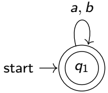
M_6 accetta tutte le possibili stringhe su $\Sigma = \{a, b\}$. Cioè

$$L(M_6) = \Sigma^*$$

Esempio 6

Sia M_6 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_6?



M_6 accetta tutte le possibili stringhe su $\Sigma = \{a, b\}$. Cioè

$$L(M_6) = \Sigma^*$$

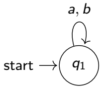
Osservazione

Ogni DFA tale che tutti gli stati sono accettanti riconosce il linguaggio Σ^*.

Esempio 7

Sia M_7 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

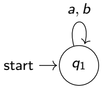
Quale linguaggio riconosce M_7?



Esempio 7

Sia M_7 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_7?



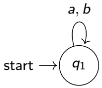
In questo caso, l'insieme degli stati accettanti è vuoto. M_7 rifiuta tutte le stringhe. Cioè

$$L(M_7) = \emptyset.$$

Esempio 7

Sia M_7 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_7?



In questo caso, l'insieme degli stati accettanti è vuoto. M_7 rifiuta tutte le stringhe. Cioè

$$L(M_7) = \emptyset.$$

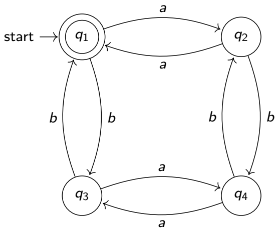
Osservazione

Ogni DFA che non ha stati accettanti rifiuta tutte le stringhe: riconosce il linguaggio $\emptyset$.

Esempio 8

Sia M_8 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

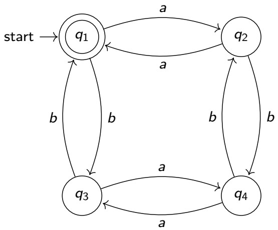
Quale linguaggio riconosce M_8?



Esempio 8

Sia M_8 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_8?

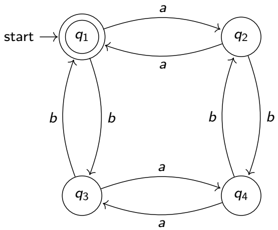


- ogni a muove da destra a sinistra o viceversa
- ogni b muove dall'alto al basso o viceversa

Esempio 8

Sia M_8 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_8?



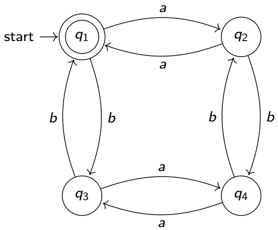
- ogni a muove da destra a sinistra o viceversa
- ogni b muove dall'alto al basso o viceversa

Per tornare su q_1 è necessario un numero pari di spostamenti orizzontali e un numero pari di spostamenti verticali.

Esempio 8

Sia M_8 l'automa con alfabeto $\Sigma = \{a, b\}$ rappresentato dal seguente diagramma di stato.

Quale linguaggio riconosce M_8?



- ogni a muove da destra a sinistra o viceversa
- ogni b muove dall'alto al basso o viceversa

Per tornare su q_1 è necessario un numero pari di spostamenti orizzontali e un numero pari di spostamenti verticali.

Il DFA M_8 riconosce il linguaggio di stringhe su Σ con numero pari di a e numero pari di b.

Fornire un DFA per ciascuno dei seguenti linguaggi sull'alfabeto $\Sigma = \{0, 1\}$:

- insieme di tutte le stringhe che terminano con 00;
- insieme di tutte le stringhe con tre zeri consecutivi;
- insieme delle stringhe con 011 come sottostringa;
- insieme delle stringhe che cominciano, finiscono, o entrambe le cose, con 01.

Operazioni regolari

Definizione

Siano A e B linguaggi. Definiamo le operazioni regolari **unione**, **concatenazione** *e* **star** *(o* **kleene star***) come segue:*

- **unione:** $A \cup B = \{x \mid x \in A \text{ o } y \in B\}$;
- **concatenazione:** $A \circ B = \{xy \mid x \in A \text{ e } y \in B\}$;
- **star:** $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0 \text{ e ogni } x_i \in A\}$.

Esempio

Sia $\Sigma = \{a, b, c, \ldots, z\}$ l'alfabeto latino di 26 lettere. Supponiamo che $A = \{\texttt{good}, \texttt{bad}\}$ e $B = \{\texttt{boy}, \texttt{girl}\}$

- **unione:** $A \cup B = \{\texttt{good, bad, boy, girl}\}$;
- **concatenazione:** $A \circ B = \{\texttt{goodboy, goodgirl, badboy, badgirl}\}$;
- **star:** $A^* = \{\epsilon, \texttt{ good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood}, \ldots\}$.

Chiusura

Una collezione S di oggetti è chiusa per un'operazione f se applicando f a membri di S, f restituisce un oggetto ancora in S.
Ad esempio $N = \{0, 1, 2, \ldots\}$ è chiuso per addizione, ma non per sottrazione.

Abbiamo visto che dato un DFA M_1 che riconosce il linguaggio L, possiamo costruire un DFA M_2 che riconosce il linguaggio complemento $L' = C(L)$:

- gli stati accettanti in M_1 diventano non accettanti in M_2.
- gli stati non accettanti in M_1 diventano accettanti in M_2.

Quindi L regolare $\implies$ $C(L)$ regolare.

Linguaggi regolari chiusi per complemento

Teorema

L'insieme dei linguaggi regolari è chiuso per l'operazione di complemento.

Dimostrazione. Sia L un linguaggio regolare su un alfabeto Σ. Esiste un DFA $M = (Q, \Sigma, f, q_1, F)$ che riconosce L.

Il DFA $M' = (Q, \Sigma, f, q_1, Q - F)$ riconosce $L' = C(L)$.

Perché funziona?

Linguaggi regolari chiusi per complemento

Teorema

L'insieme dei linguaggi regolari è chiuso per l'operazione di complemento.

Dimostrazione. Sia L un linguaggio regolare su un alfabeto Σ. Esiste un DFA $M = (Q, \Sigma, f, q_1, F)$ che riconosce L.

Il DFA $M' = (Q, \Sigma, f, q_1, Q - F)$ riconosce $L' = C(L)$.

Perché funziona?
Ogni stringa in L è accettata da M (M termina in uno stato in $q \in F$) e quindi rifiutata da M' (perché $q \notin Q - F$).

Ogni stringa in $C(L)$ è rifiutata da M e quindi è accettata da M'.

Linguaggi regolari chiusi per complemento

Teorema

L'insieme dei linguaggi regolari è chiuso per l'operazione di complemento.

Dimostrazione. Sia L un linguaggio regolare su un alfabeto Σ. Esiste un DFA $M = (Q, \Sigma, f, q_1, F)$ che riconosce L.

Il DFA $M' = (Q, \Sigma, f, q_1, Q - F)$ riconosce $L' = C(L)$.

Perché funziona?
Ogni stringa in L è accettata da M (M termina in uno stato in $q \in F$) e quindi rifiutata da M' (perché $q \notin Q - F$).

Ogni stringa in $C(L)$ è rifiutata da M e quindi è accettata da M'.

$$M' \text{ accetta } x \iff x \in C(L).$$

Linguaggi regolari chiusi per unione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di unione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cup L_2$.

Dimostrazione (idea).

Linguaggi regolari chiusi per unione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di unione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cup L_2$.

Dimostrazione (idea). L_1 ha un DFA M_1. L_2 ha un DFA M_2. Una stringa w è in $L_1 \cup L_2$ se e solo se w è accettata da M_1 oppure da M_2. Bisogna definire un DFA M_3 che accetti w se e solo se w è accettata da M_1 o M_2.

Linguaggi regolari chiusi per unione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di unione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cup L_2$.

Dimostrazione (idea). L_1 ha un DFA M_1. L_2 ha un DFA M_2. Una stringa w è in $L_1 \cup L_2$ se e solo se w è accettata da M_1 oppure da M_2. Bisogna definire un DFA M_3 che accetti w se e solo se w è accettata da M_1 o M_2.

M_3 dovrà essere capace di

- tener traccia di dove l'input sarebbe se fosse contemporaneamente in input a M_1 e M_2;
- accettare una stringa w se e solo se M_1 oppure M_2 accettano la stringa.

Linguaggi regolari chiusi per unione

Siano L_1 e L_2 definiti sullo stesso alfabeto Σ (non si perde di generalità).
Sia $M_1 = (Q_1, \Sigma, f_1, q_1, F_1)$ il DFA che riconosce L_1.
Sia $M_2 = (Q_2, \Sigma, f_2, q_2, F_2)$ il DFA che riconosce L_2.

Costruiamo il DFA $M_3 = (Q_3, \Sigma, f_3, q_3, F_3)$:

- $Q_3 = Q_1 \times Q_2 = \{(x, y) | \; x \in Q_1, y \in Q_2\}$
- L'alfabeto è lo stesso: Σ
- $f_3 : Q_3 \times \Sigma \to Q_3$
  per ogni $x \in Q_1$, $y \in Q_2$, e ogni $a \in \Sigma$: $f_3((x, y), a) = (f_1(x, a), f_2(y, a))$
- q_3 è la coppia (q_1, q_2)
- $F_3 = \{(x, y) \in Q_3 | \; x \in F_1 \text{ o } y \in F_2\}$

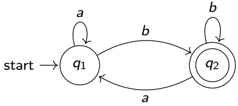
Linguaggi regolari chiusi per unione

M_3 è un DFA poiché il numero di stati in M_3 è finito: $|Q3| = |Q1| \cdot |Q2|$ (Q_1 e Q_2 sono insiemi finiti).

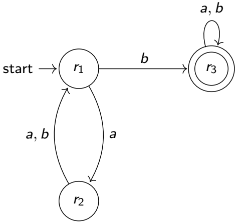
M_3 riconosce $L_1 \cup L_2$: ogni stringa $x \in \Sigma^*$ è accettata da M_1 o da M_2 se e solo se esiste una sequenza di stati di M_3 che termina in uno stato in F_3 (completare i dettagli per esercizio).

Esempio

Si considerino due linguaggi regolari L_1 e L_2 su $\Sigma = \{a, b\}$.
DFA M_1 riconosce linguaggio $L_1 = L(M_1)$
DFA M_2 riconosce linguaggio $L_2 = L(M_2)$



M_1

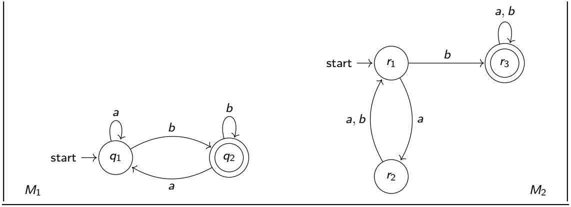


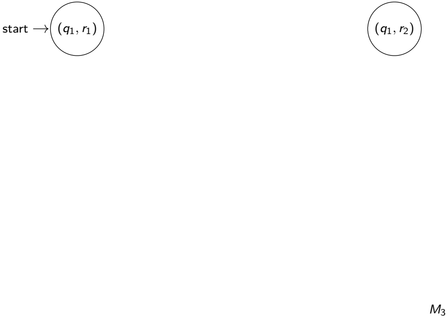
M_2

Costruiamo il DFA $M_3 = (Q_3, \Sigma, \delta, q_3, F_3)$ che riconosce $L(M_1) \cup L(M_2)$.

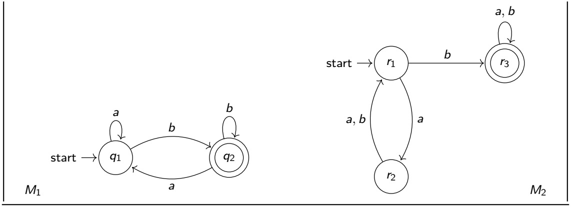
$Q_3 = \{(q_1, r_1), (q_1, r_2), (q_1, r_3), (q_2, r_1), (q_2, r_2), (q_2, r_3)\}$.

Esempio

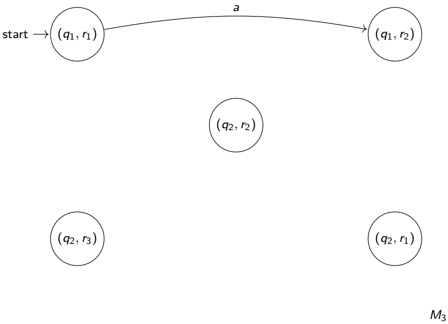




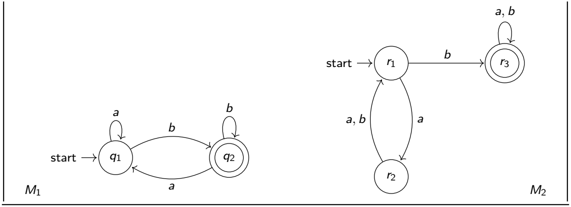
Esempio



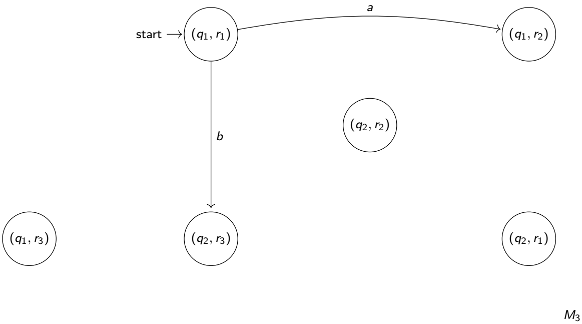
$$f_3((q_1, r_1), a) = (f_1(q_1, a), f_2(r_1, a)) = (q_1, r_2)$$



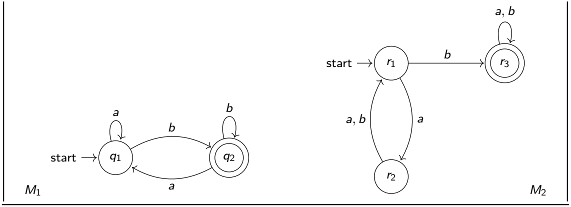
Esempio



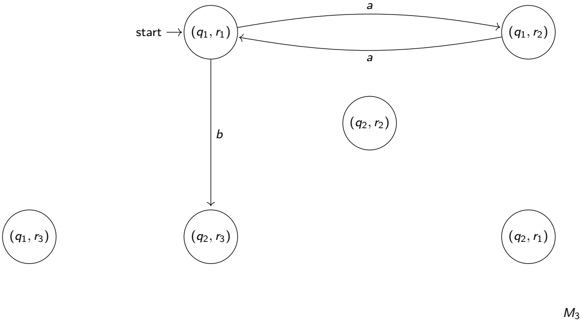
$f_3((q_1, r_1), b) = (f_1(q_1, b), f_2(r_1, b)) = (q_2, r_3)$



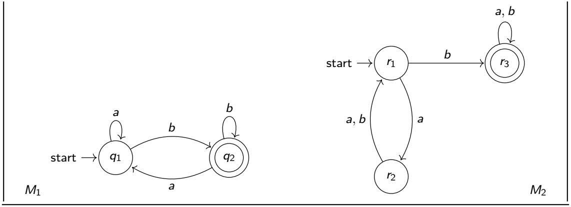
Esempio



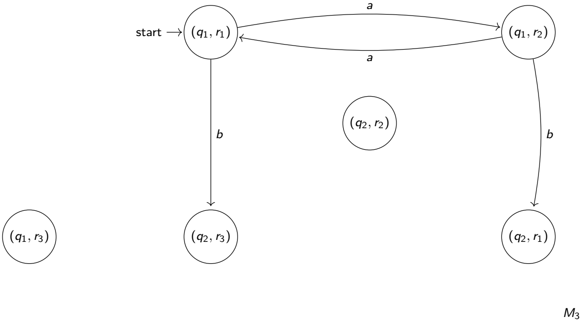
$$f_3((q_1, r_2), a) = (f_1(q_1, a), f_2(r_2, a)) = (q_1, r_1)$$



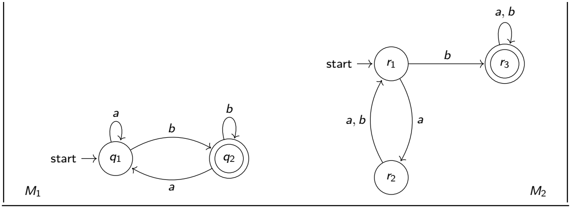
Esempio



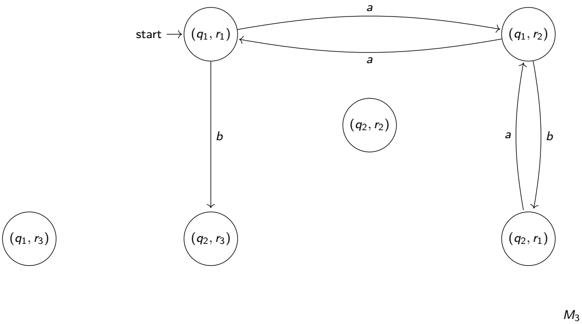
$$f_3((q_1, r_2), b) = (f_1(q_1, b), f_2(r_2, b)) = (q_2, r_1)$$



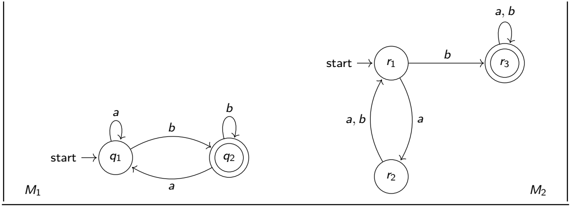
Esempio



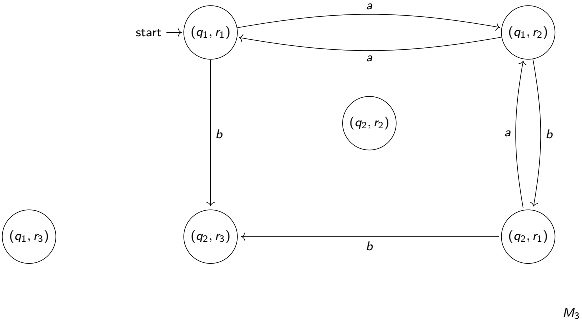
$f_3((q_2, r_1), a) = (f_1(q_2, a), f_2(r_1, a)) = (q_1, r_2)$



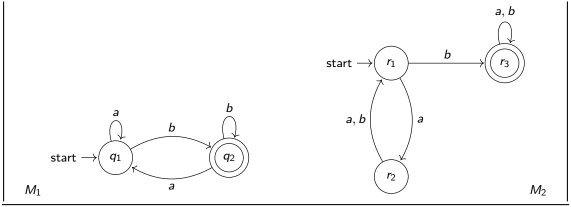
Esempio



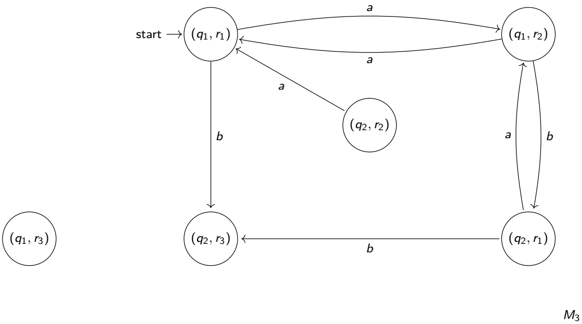
$f_3((q_2, r_1), b) = (f_1(q_2, b), f_2(r_1, b)) = (q_2, r_3)$



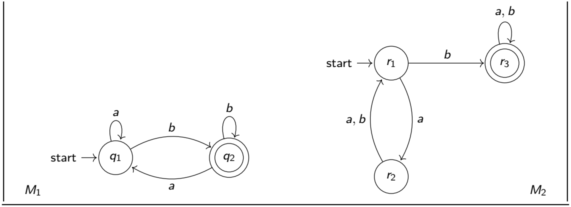
Esempio



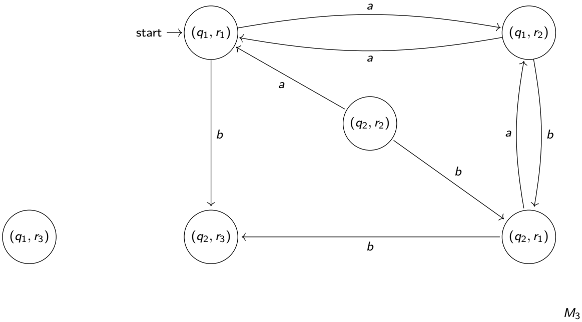
$$f_3((q_2, r_2), a) = (f_1(q_2, a), f_2(r_2, a)) = (q_1, r_1)$$



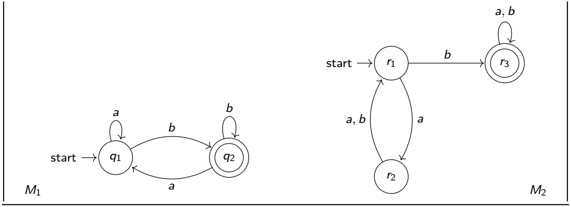
Esempio



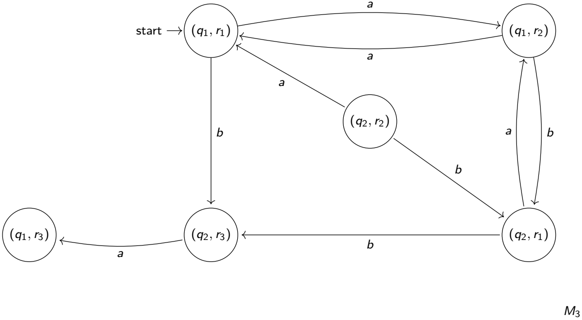
$f_3((q_2, r_2), b) = (f_1(q_2, b), f_2(r_2, b)) = (q_2, r_1)$



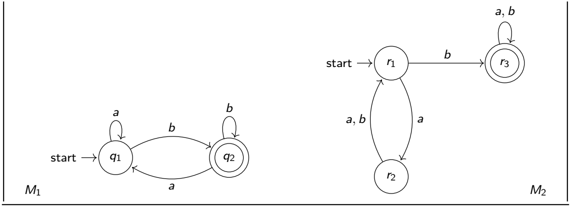
Esempio



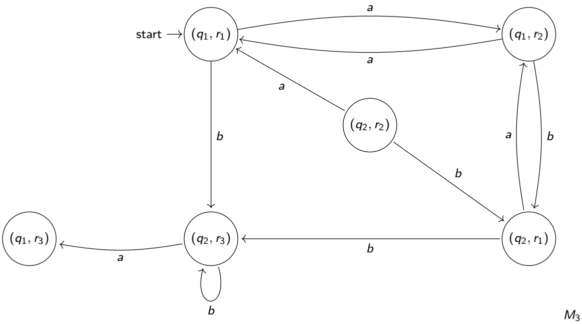
$$f_3((q_2, r_3), a) = (f_1(q_2, a), f_2(r_3, a)) = (q_1, r_3)$$



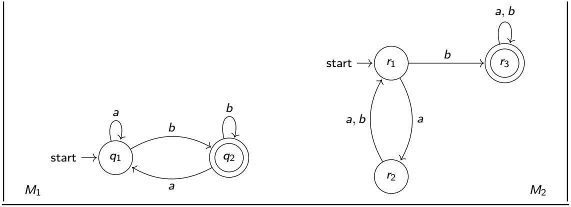
Esempio



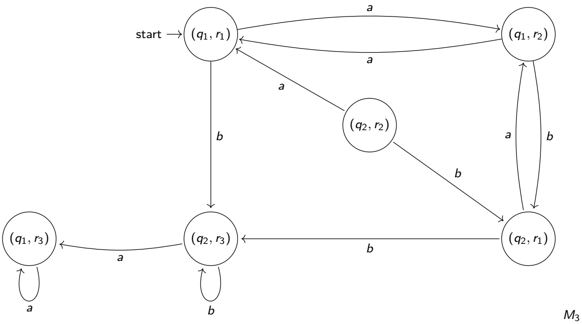
$f_3((q_2, r_3), b) = (f_1(q_2, b), f_2(r_3, b)) = (q_2, r_3)$



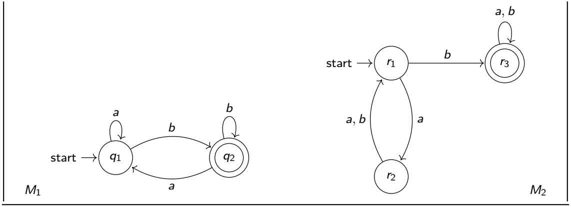
Esempio



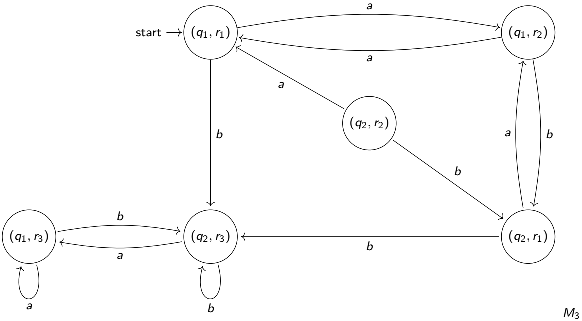
$$f_3((q_1, r_3), a) = (f_1(q_1, a), f_2(r_3, a)) = (q_1, r_3)$$



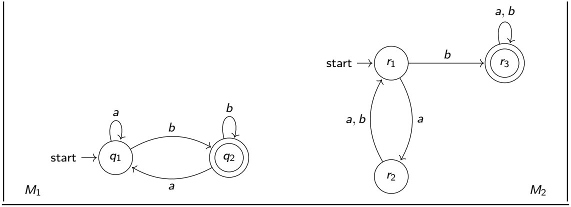
Esempio



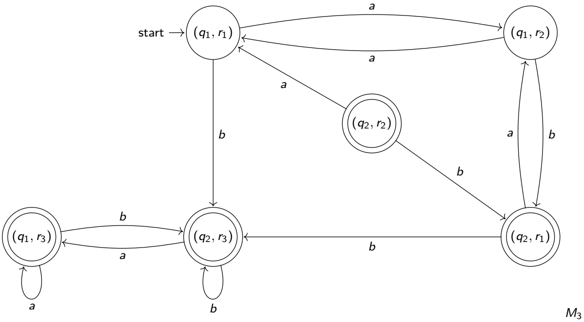
$f_3((q_1, r_3), b) = (f_1(q_1, b), f_2(r_3, b)) = (q_2, r_3)$



Esempio



stati accettanti: $F_3 = \{(q_2, r_1), (q_2, r_2), (q_2, r_3), (q_1, r_3)\}$



Linguaggi regolari chiusi per intersezione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di intersezione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cap L_2$.

Dimostrazione (idea).

Linguaggi regolari chiusi per intersezione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di intersezione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cap L_2$.

Dimostrazione (idea). L_1 ha un DFA M_1. L_2 ha un DFA M_2. Una stringa w è in $L_1 \cap L_2$ se e solo se w è accettata sia da M_1 sia da M_2. Bisogna definire un DFA M_3 che accetti w se e solo se w è accettata da entrambe le macchine M_1 e M_2.

Linguaggi regolari chiusi per intersezione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di intersezione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cap L_2$.

Dimostrazione (idea). L_1 ha un DFA M_1. L_2 ha un DFA M_2. Una stringa w è in $L_1 \cap L_2$ se e solo se w è accettata sia da M_1 sia da M_2. Bisogna definire un DFA M_3 che accetti w se e solo se w è accettata da entrambe le macchine M_1 e M_2.

M_3 dovrà essere capace di

- tener traccia di dove l'input sarebbe se fosse contemporaneamente in input a M_1 e M_2;
- accettare una stringa w se e solo se sia M_1 sia M_2 accettano la stringa.

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di intersezione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \cap L_2$.

Dimostrazione (idea). L_1 ha un DFA M_1. L_2 ha un DFA M_2. Una stringa w è in $L_1 \cap L_2$ se e solo se w è accettata sia da M_1 sia da M_2. Bisogna definire un DFA M_3 che accetti w se e solo se w è accettata da entrambe le macchine M_1 e M_2.

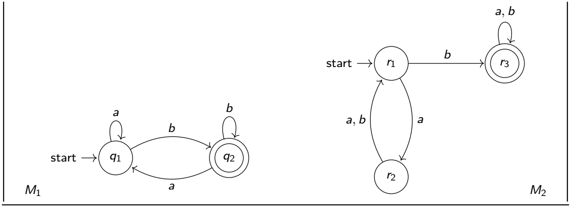
M_3 dovrà essere capace di

- tener traccia di dove l'input sarebbe se fosse contemporaneamente in input a M_1 e M_2;
- accettare una stringa w se e solo se sia M_1 sia M_2 accettano la stringa.

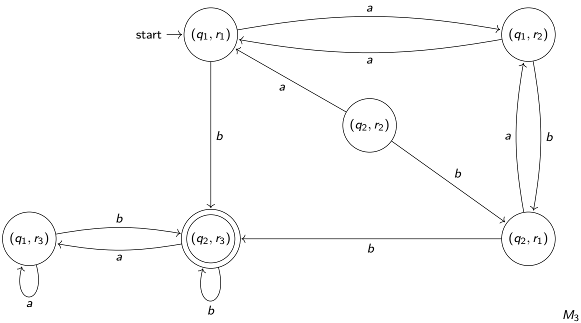
Occorre definire un DFA M_3 che accetta w se e solo se w è accettata da M_1 e M_2.

- Fornire la definizione formale di M_3
- Mostrare che M_3 accetta w se e solo se w è accettata sia da M_1 che da M_2.

Esempio



start
q_1
q_2
a
b
b
a
M_1
start
r_1
r_2
r_3
b
a, b
a, b
a
M_2



start
(q_1, r_1)
(q_1, r_2)
(q_2, r_2)
(q_1, r_3)
(q_2, r_3)
(q_2, r_1)
a
a
a
b
b
a
b
b
a
b
a
b
M_3

Linguaggi regolari chiusi per concatenazione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

Come dimostrarlo? Si potrebbe procedere come fatto finora: partire da un DFA M_1 che riconosce L_1 e un DFA M_2 che riconosce L_2 e costruire un DFA M_3 che riconosca $L_1 \circ L_2$.

Come costruire M_3? L'automa M_3 dovrebbe accettare una stringa w se e solo se essa può essere divisa in due parti tali che la prima parte è accettata da M_1 e la seconda parte è accettata da M_2.

Il problema è che M_3 non sa dove finisce la prima parte e dove comincia la seconda. Si dovrebbero analizzare tutte le possibilità: troppo laborioso!

Linguaggi regolari chiusi per concatenazione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

Come dimostrarlo? Si potrebbe procedere come fatto finora: partire da un DFA M_1 che riconosce L_1 e un DFA M_2 che riconosce L_2 e costruire un DFA M_3 che riconosca $L_1 \circ L_2$.

Come costruire M_3? L'automa M_3 dovrebbe accettare una stringa w se e solo se essa può essere divisa in due parti tali che la prima parte è accettata da M_1 e la seconda parte è accettata da M_2.

Il problema è che M_3 non sa dove finisce la prima parte e dove comincia la seconda. Si dovrebbero analizzare tutte le possibilità: troppo laborioso!

Lasciamo per il momento in sospeso il problema e introduciamo un nuovo argomento: il **non determinismo**.