Computabilità, Complessità e Logica

Lezione 3

Che linguaggi sono regolari?

Alcune costruzioni semplici

- $L = \emptyset$ e $L = \Sigma^\star$ sono due linguaggi regolari (gli automi hanno rispettivamente nessuno e tutti gli stati accettanti)
- I linguaggi contenenti una sola lettera
- Dati due linguaggi regolari come possiamo combinarli per ottenere un altro linguaggio regolare?

Proprietà di chiusura

La classe dei linguaggi regolari è chiusa rispetto alle comuni operazioni insiemistiche:

1. **Unione** (finita)
   Se L_1 e L_2 sono regolari allora $L_1 \cup L_2$ è regolare

2. **Intersezione** (finita)
   Se L_1 e L_2 sono regolari allora $L_1 \cap L_2$ è regolare

3. **Complementazione**
   Se L è regolare allora $\Sigma^\star - L$ è regolare

Proprietà di chiusura

La classe dei linguaggi regolari è anche chiusa rispetto alle seguenti operazioni:

4. **Concatenazione**

   Se L_1 e L_2 sono regolari allora la loro concatenazione $L_1L_2 = \{w\colon\ w = xy$ con $x \in L_1$ e $y \in L_2\}$ è regolare

5. **Stella di Kleene**

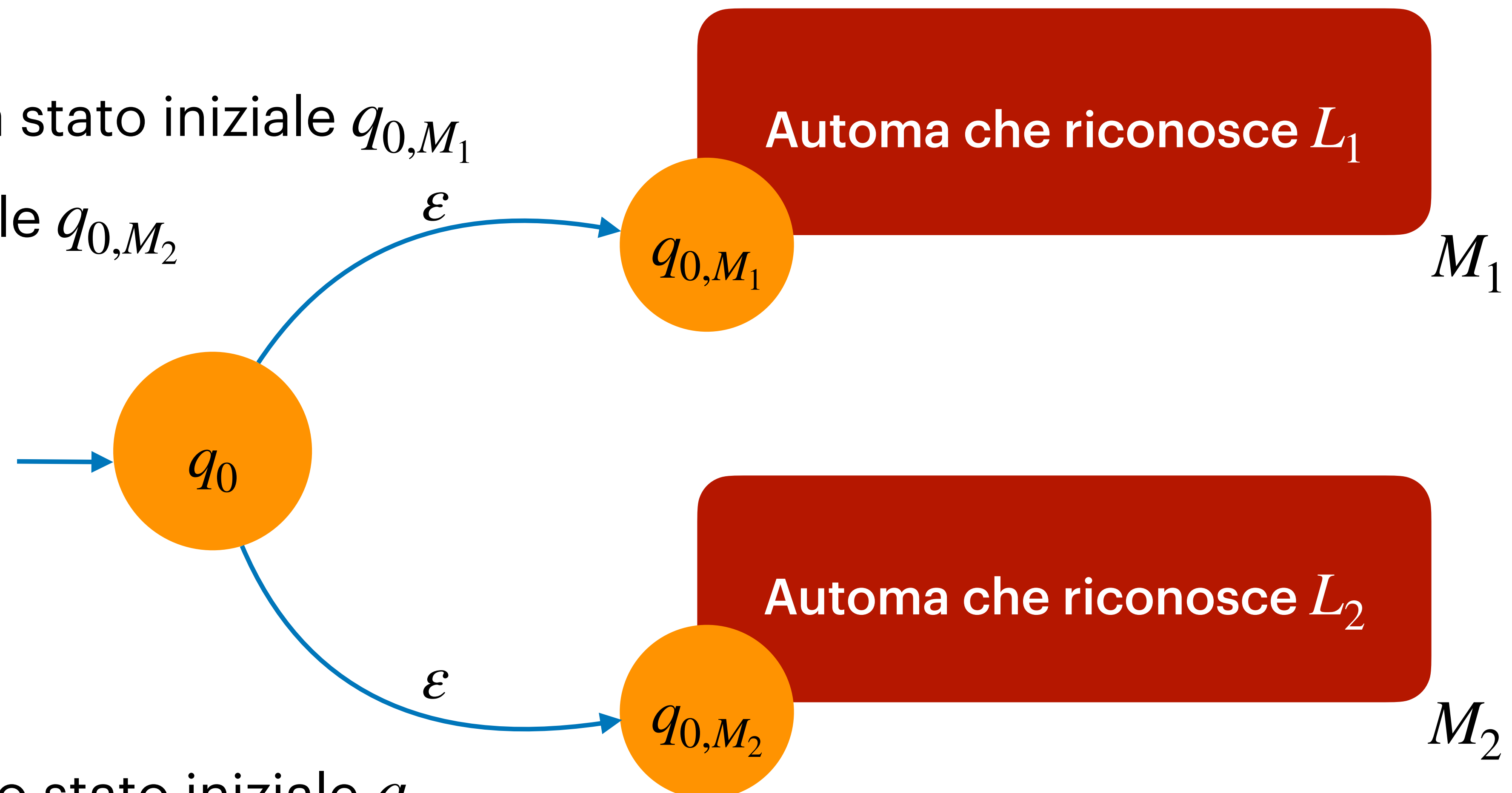
   Se L è regolare, allora $L^\star = \{x_1x_2\cdots x_k\colon\ k \in \mathbb{N}$ e $x_i \in L$ per $1 \leq i \leq k\}$ è un linguaggio regolare

Unione

Supponiamo di avere due automi M_1 e M_2 che riconoscono, rispettivamente, i linguaggi L_1 e L_2.

Come possiamo riconoscere $L_1 \cup L_2$?

Supponiamo M_1 abbia stato iniziale q_{0,M_1} e M_2 abbia stato iniziale q_{0,M_2}



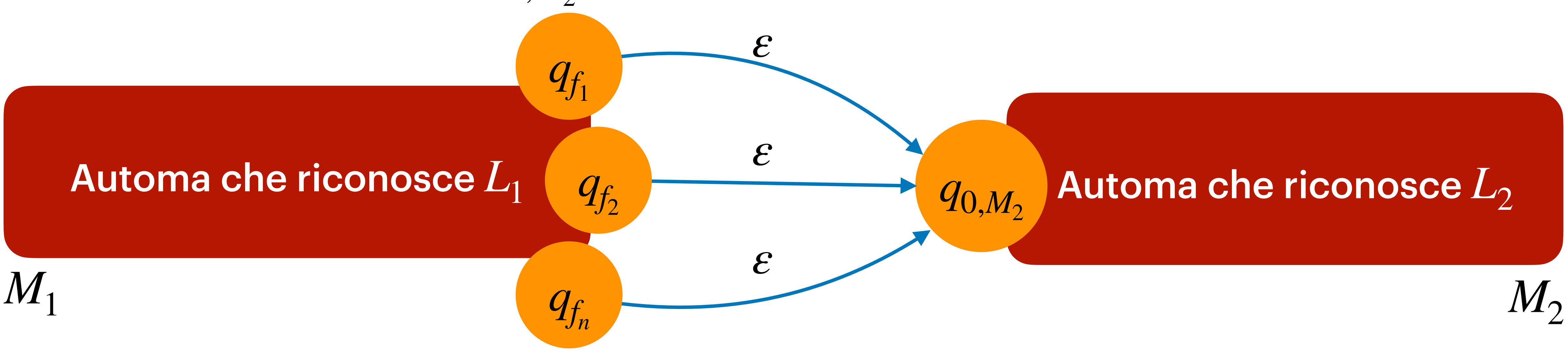
Aggiungiamo un nuovo stato iniziale q_0 che colleghiamo con ε-transizioni a q_{0,M_1} e q_{0,M_2}

Concatenazione

Supponiamo di avere due automi M_1 e M_2 che riconoscono, rispettivamente, i linguaggi L_1 e L_2.

Come possiamo riconoscere $L_1 L_2$?

Supponiamo M_1 abbia stati finali $q_{f_1}, q_{f_2}, \ldots, q_{f_m}$
e M_2 abbia stato iniziale q_{0,M_2}



Colleghiamo gli stati *finali* di M_1 allo stato iniziale di M_2 con delle ε-transizioni

Gli stati accettanti dell'automa risultante sono solo quelli di M_2

Stella di Kleene

Supponiamo di avere un automa M che riconosce il linguaggio L

Come possiamo riconoscere $L^\star$?

Supponiamo M abbia stati finali $q_{f_1}, q_{f_2}, \ldots, q_{f_m}$ e abbia stato iniziale q_0

Colleghiamo gli stati *finali* di M al suo stato iniziale con delle ε-transizioni

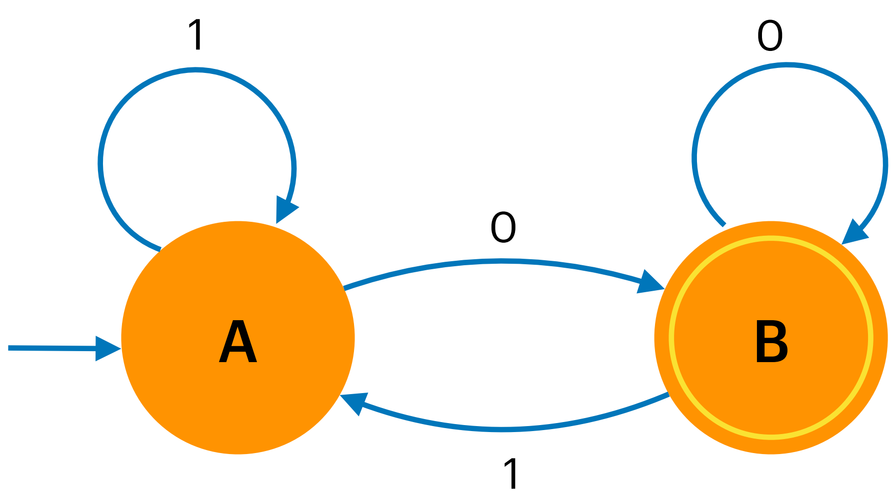
Ci manca il poter riconoscere la parola vuota: possiamo fare l'unione con $\{\varepsilon\}$

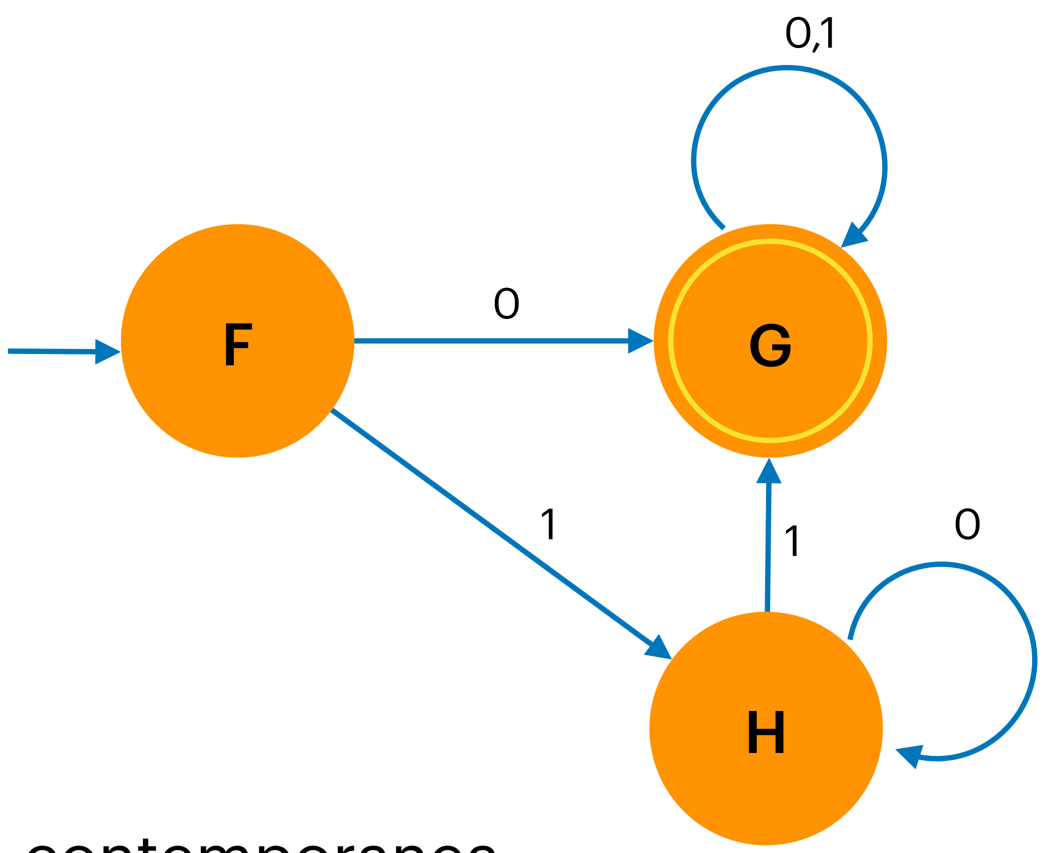
Prodotto di automi

- Anche solo con concatenazione e unione possiamo già dire che tutti i linguaggi finiti sono regolari:
  - Ogni parola è ottenuta tramite concatenazione delle sue lettere
  - L'unione di linguaggi con un numero finito di parole è regolare
- Le ε-transizioni ci aiutano nelle costruzioni per concatenazione e unione
- Per le altre costruzioni è utile definire un **prodotto** di automi a stati finiti

Prodotto di automi

Supponiamo di avere due automi M_1 e M_2 che riconoscono, rispettivamente, i linguaggi L_1 e L_2.





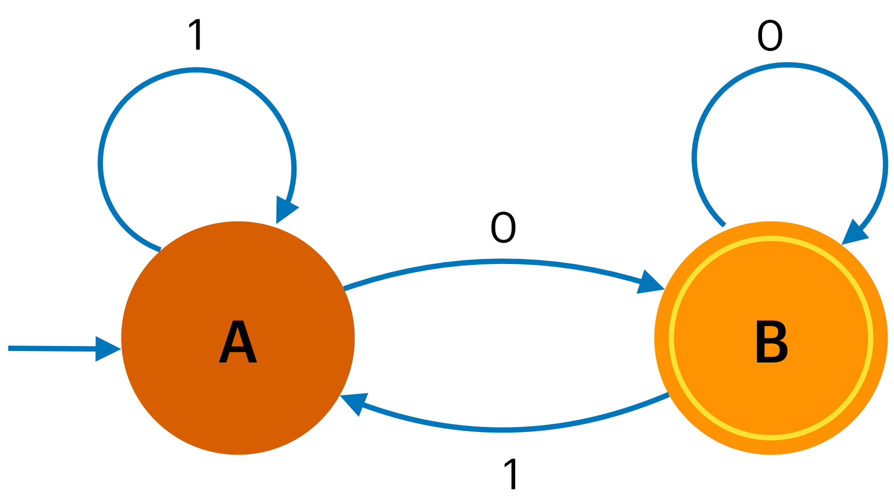
Vogliamo che questi “leggano” una parola w in contemporanea.

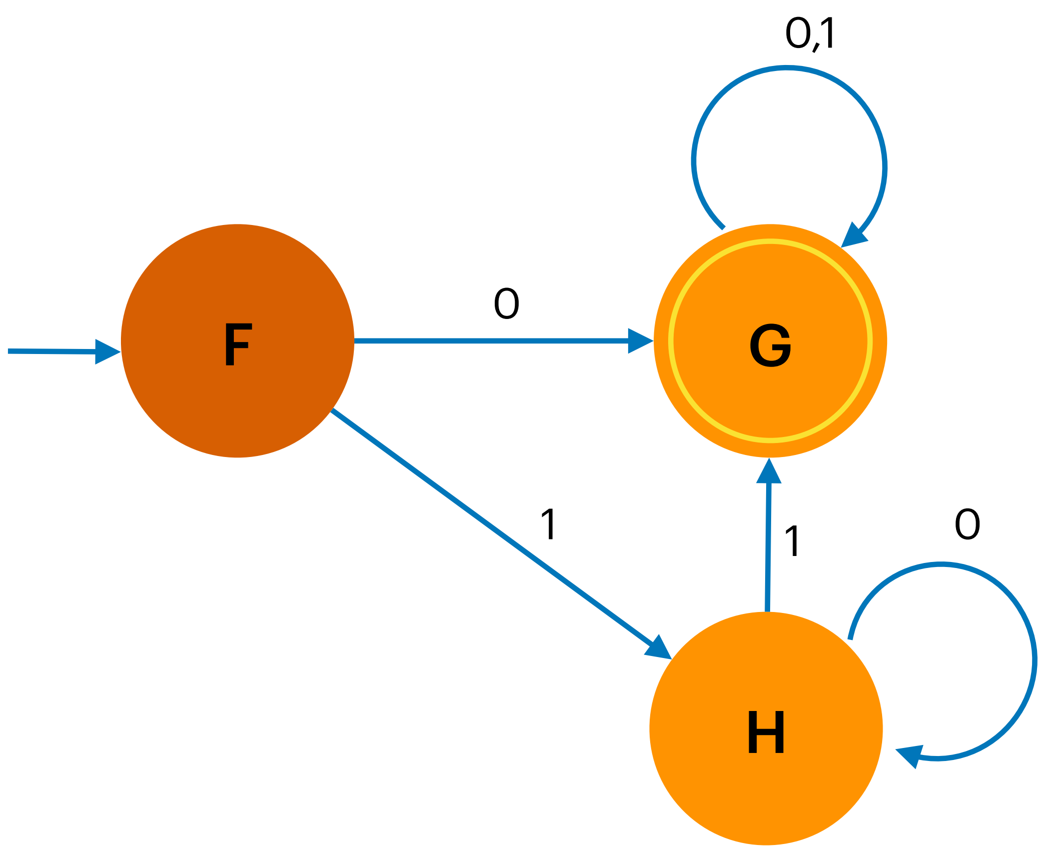
Per esempio $w = 110$

Prodotto di automi

$w = 110$

Parte letta: ε

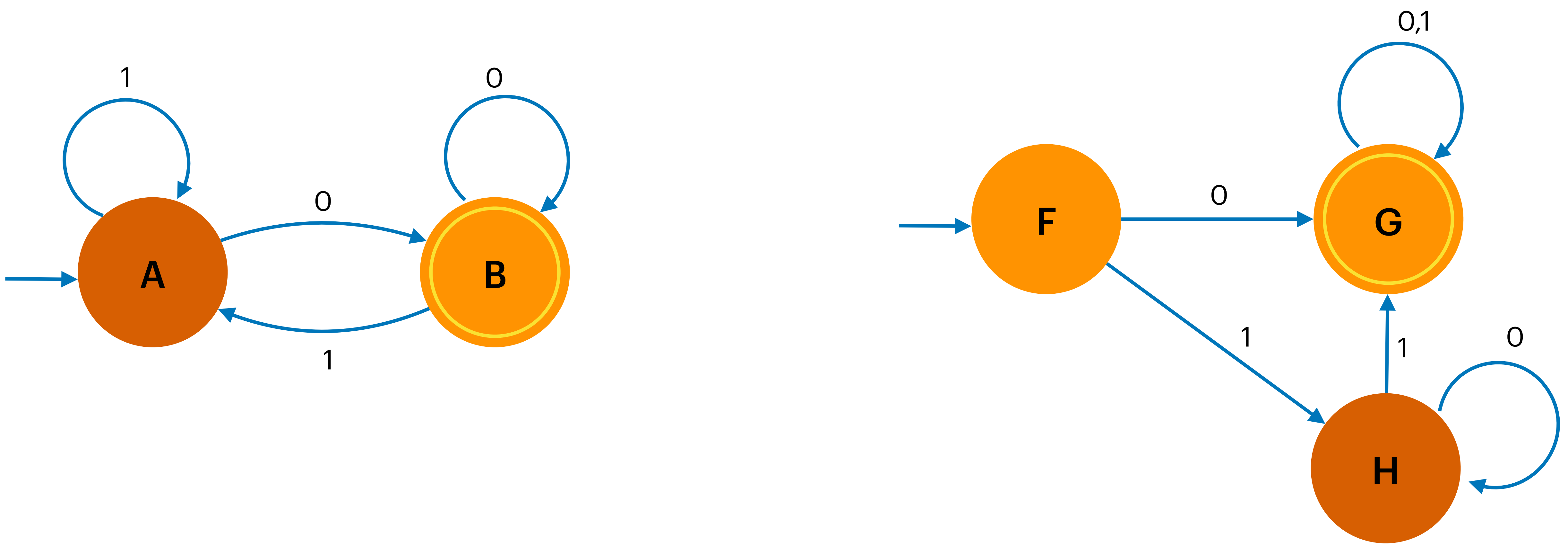




Prodotto di automi

$w = 110$

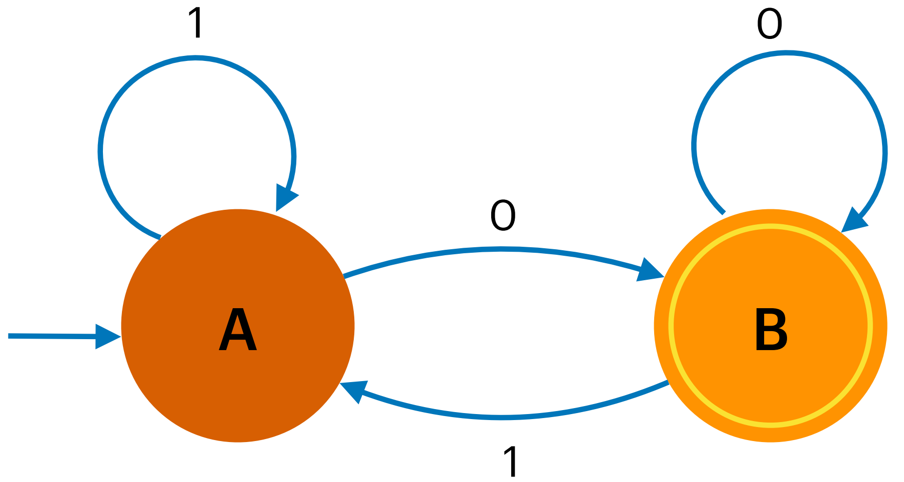
Parte letta: 1

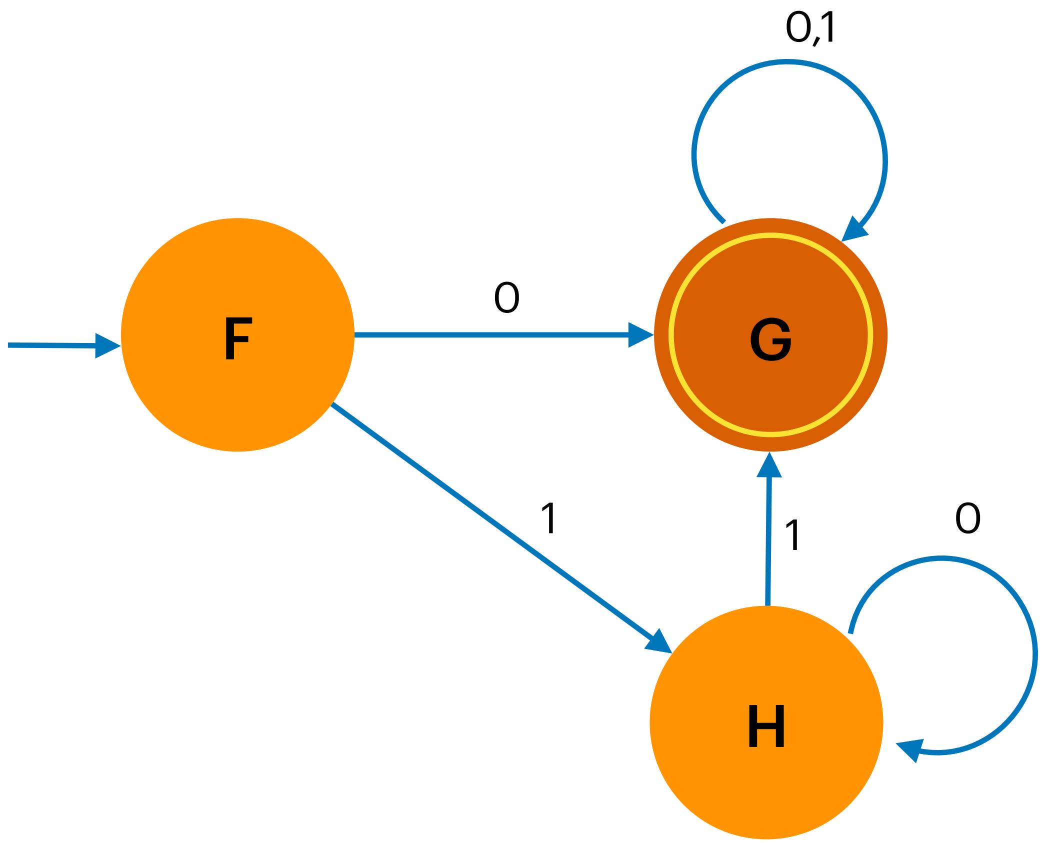


Prodotto di automi

$w = 110$

Parte letta: 11

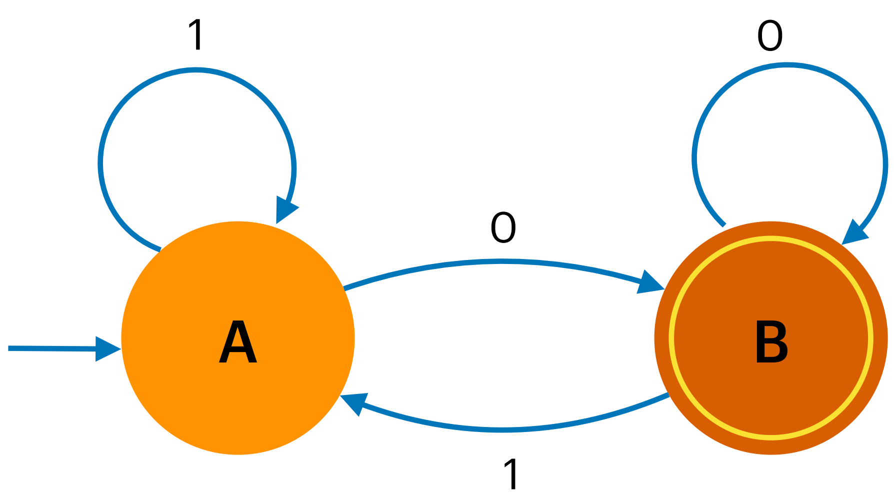


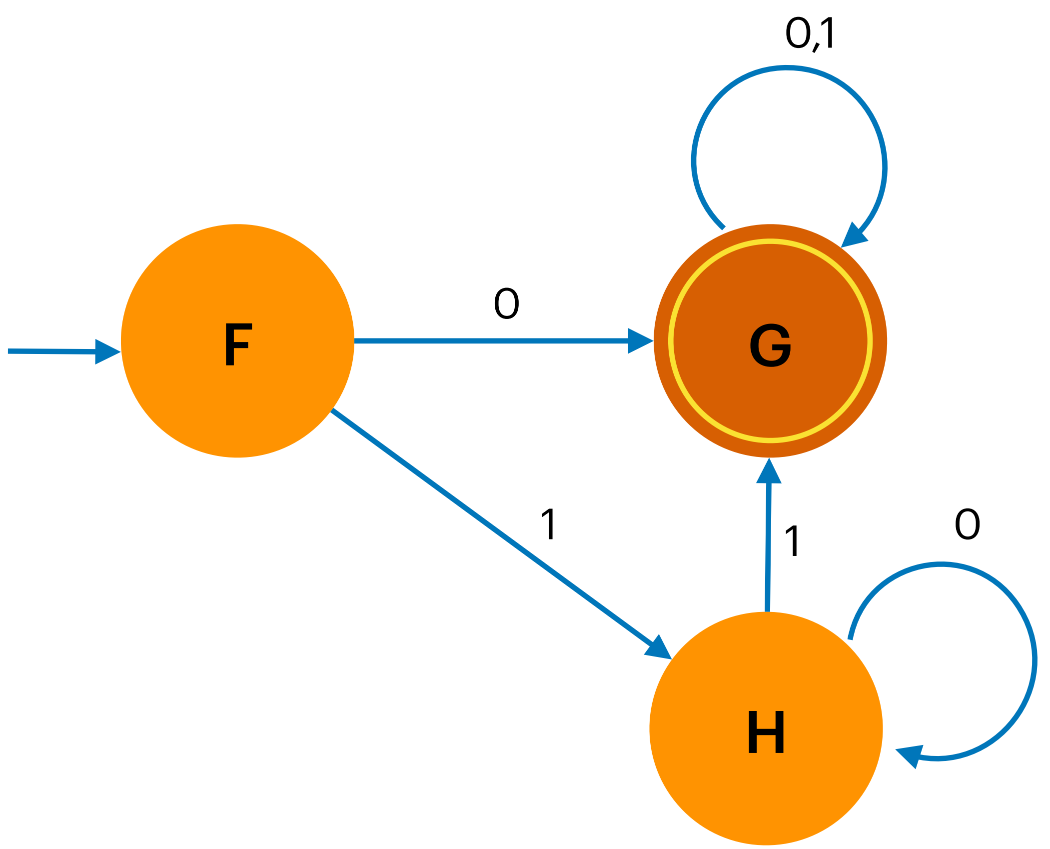


Prodotto di automi

$w = 110$

Parte letta: 110





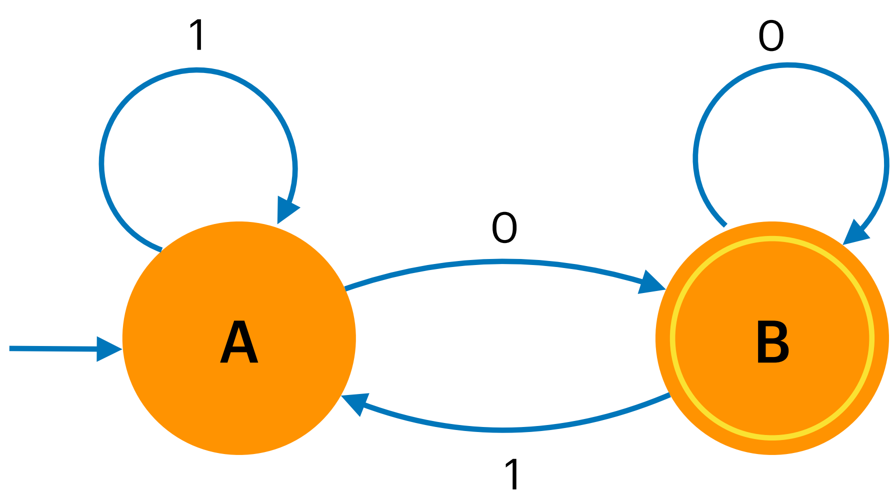
Entrambi gli automi accettano

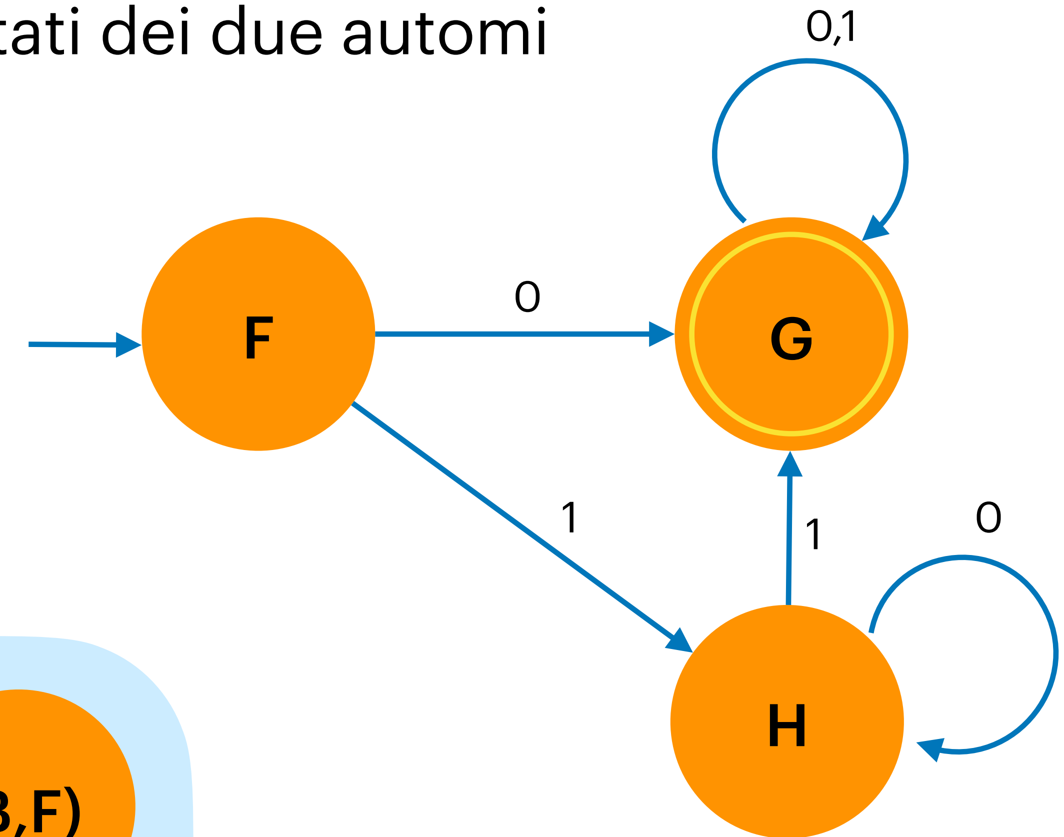
Potremmo quindi dedurre che $w \in L_1 \cap L_2$

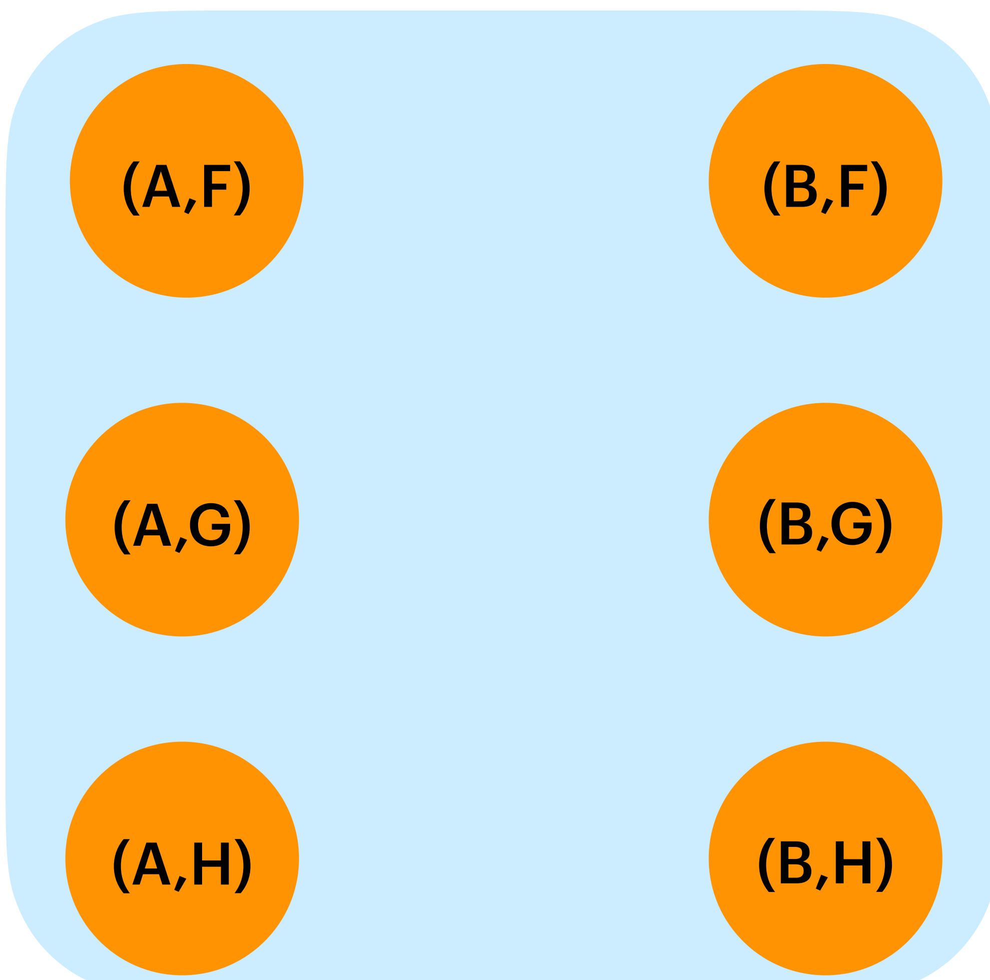
Ma come facciamo a farlo con un solo automa?

Prodotto di automi

Creiamo un automa i cui stati sono le *coppie* di stati dei due automi







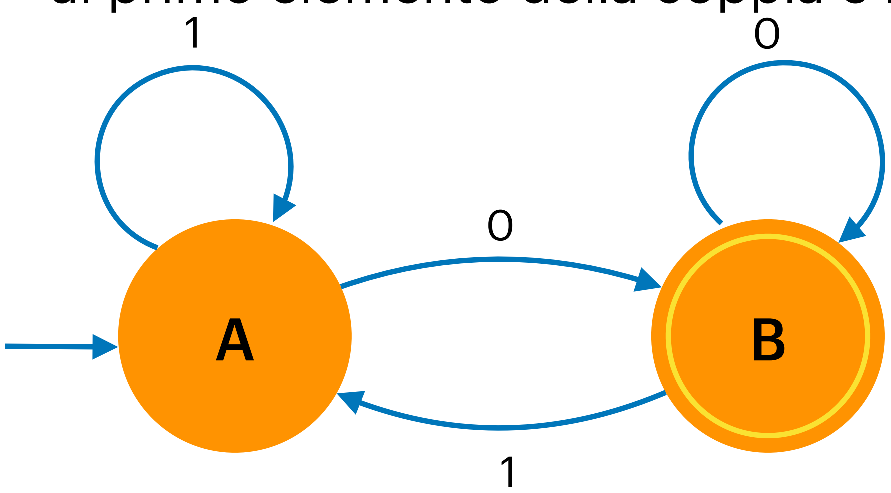
Gli stati sono $Q = Q_1 \times Q_1$

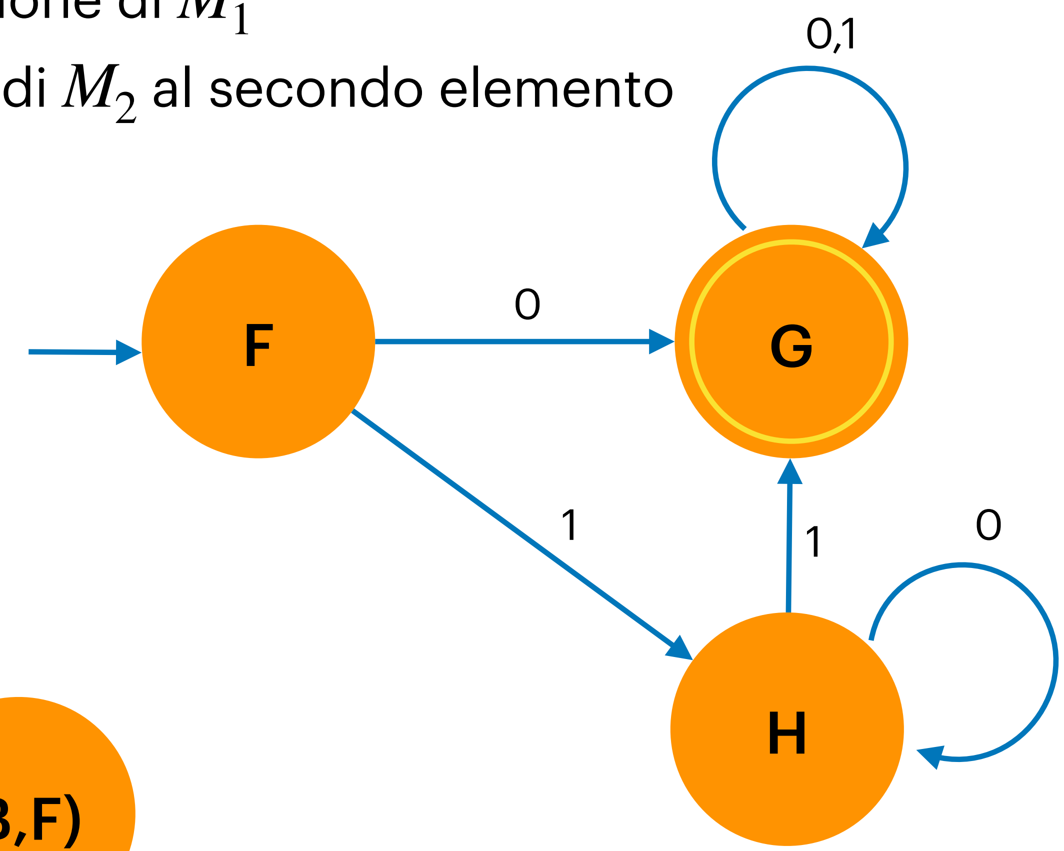
con Q_1 gli stati di M_1

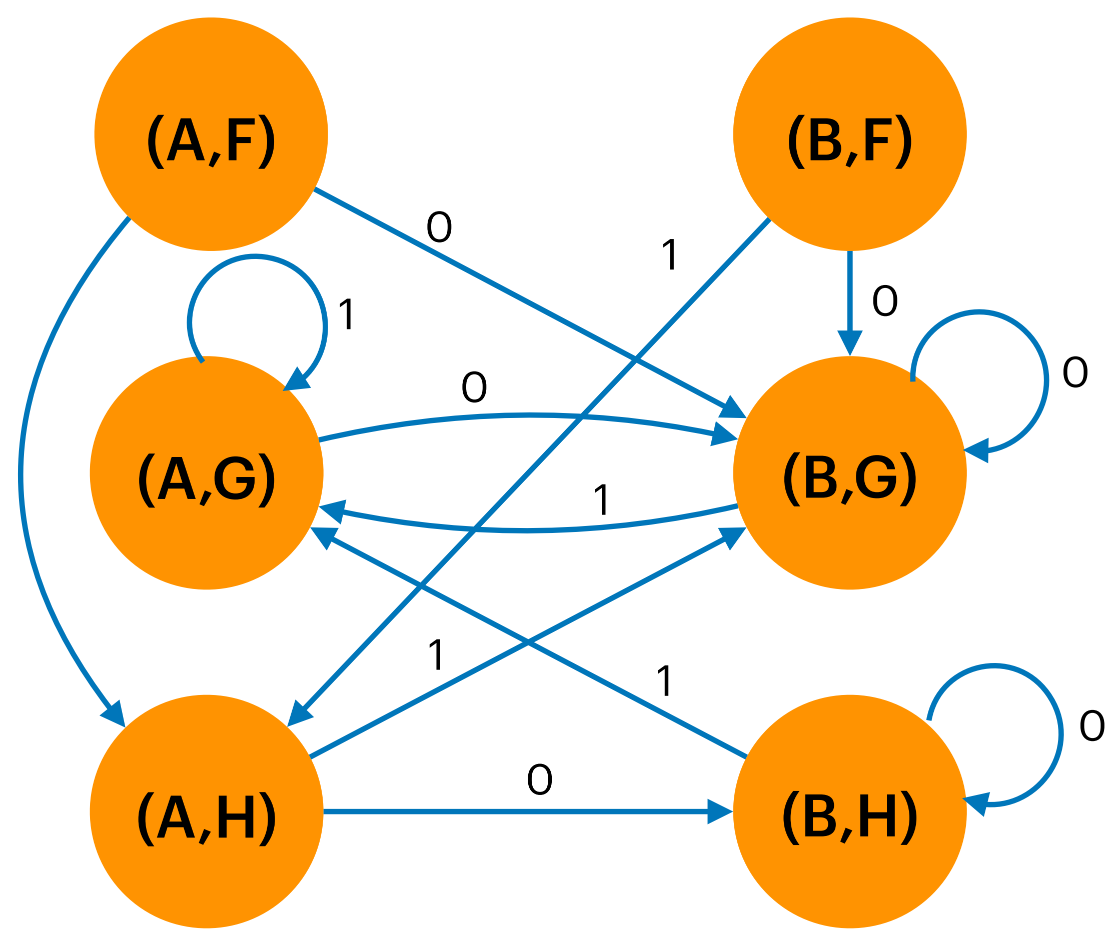
e Q_2 gli stati di M_2

Prodotto di automi

Le transizioni sono date dall'applicare la funzione di transizione di M_1
al primo elemento della coppia e la funzione di transizione di M_2 al secondo elemento





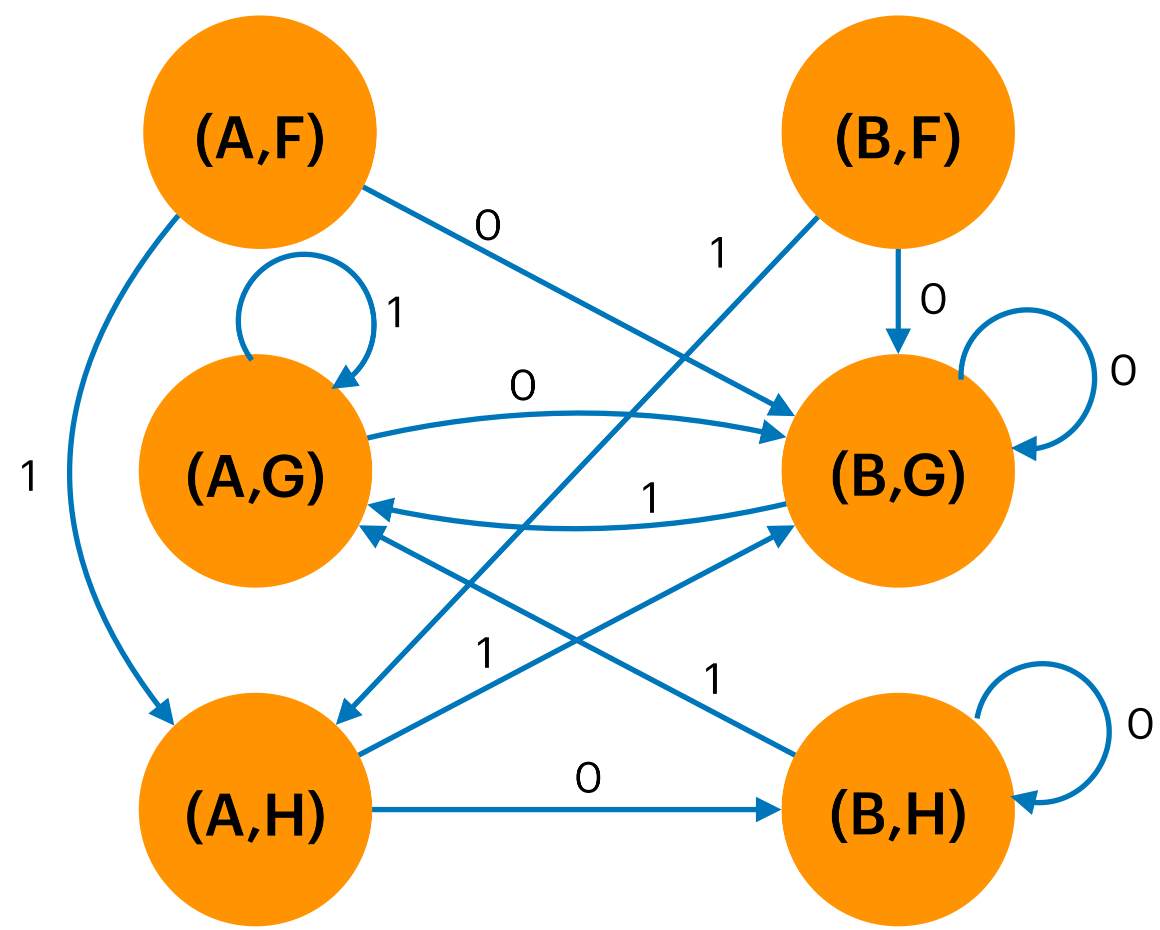


Prodotto di automi

Formalmente, se $M_1 = (Q_1, \Sigma, \delta_1, q_{M_1,1}, F_1)$ e $M_2 = (Q_2, \Sigma, \delta_2, q_{M_2,1}, F_2)$ allora la funzione di transizione dell'automa prodotto è definita come:

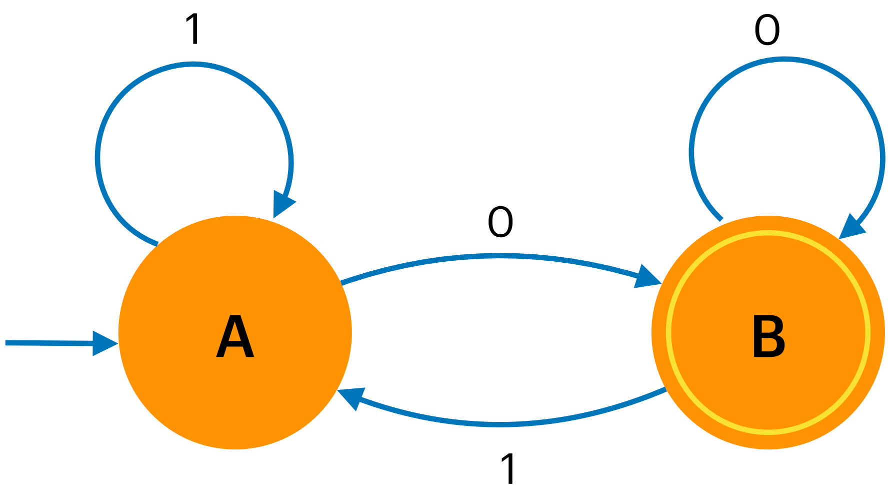
$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

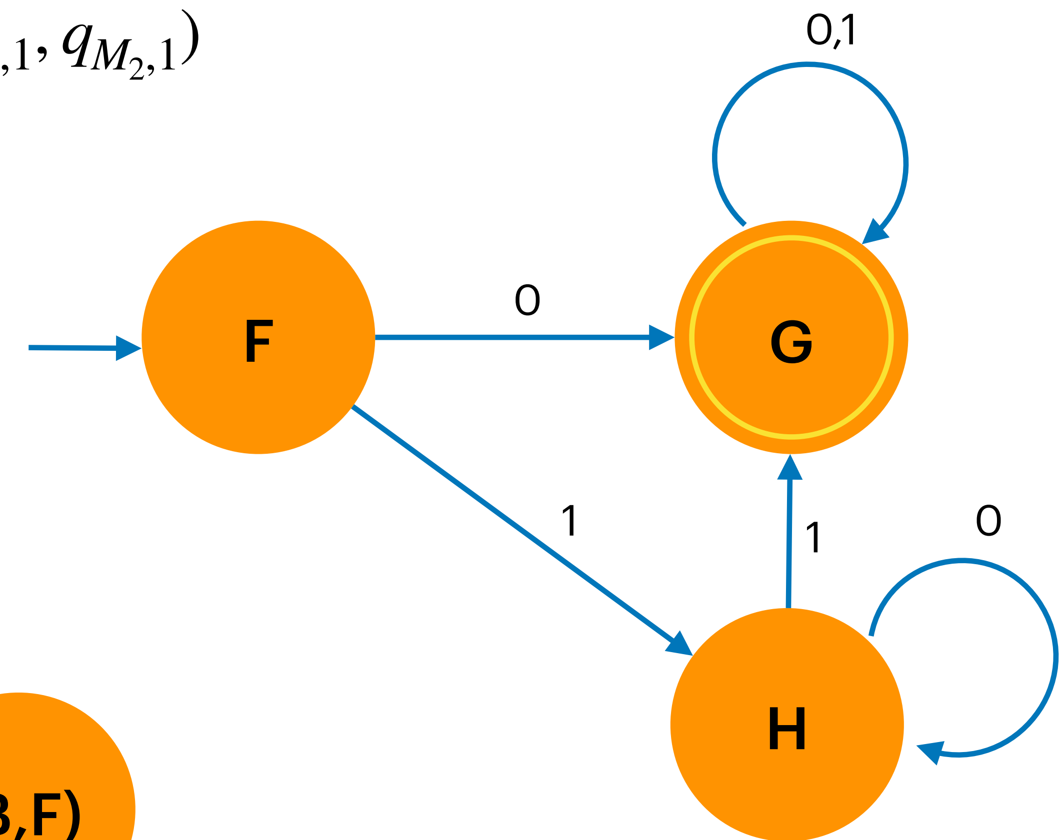
per ogni $(q_1, q_2) \in Q_1 \times Q_2$ e ogni $a \in \Sigma$

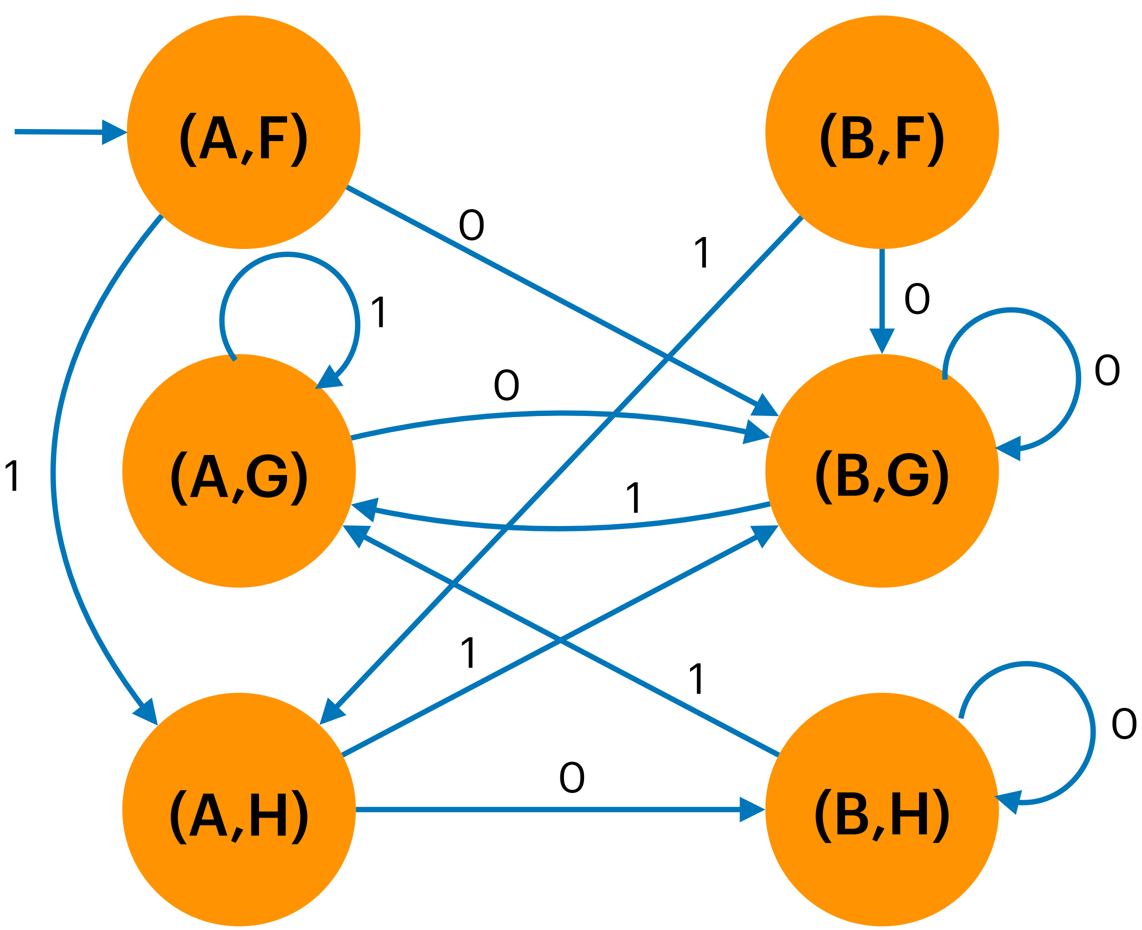


Prodotto di automi

Lo stato iniziale è lo la coppia degli stati iniziali, quindi $(q_{M_1,1}, q_{M_2,1})$

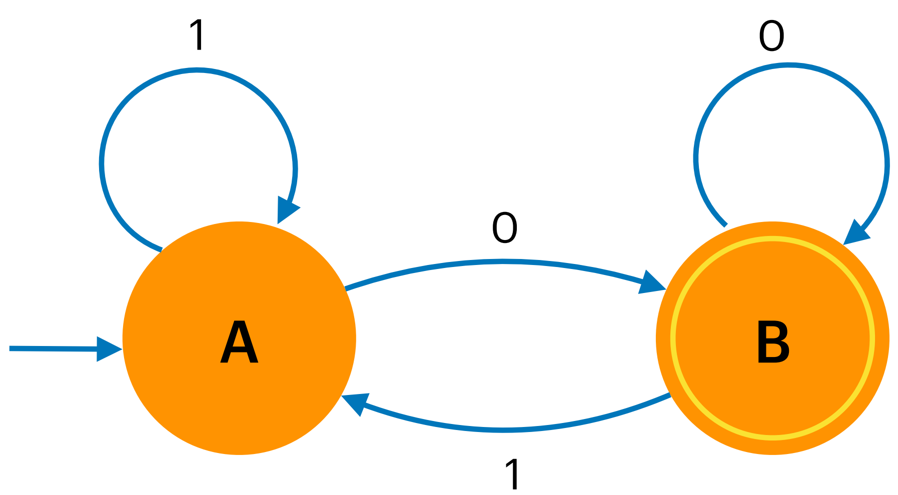


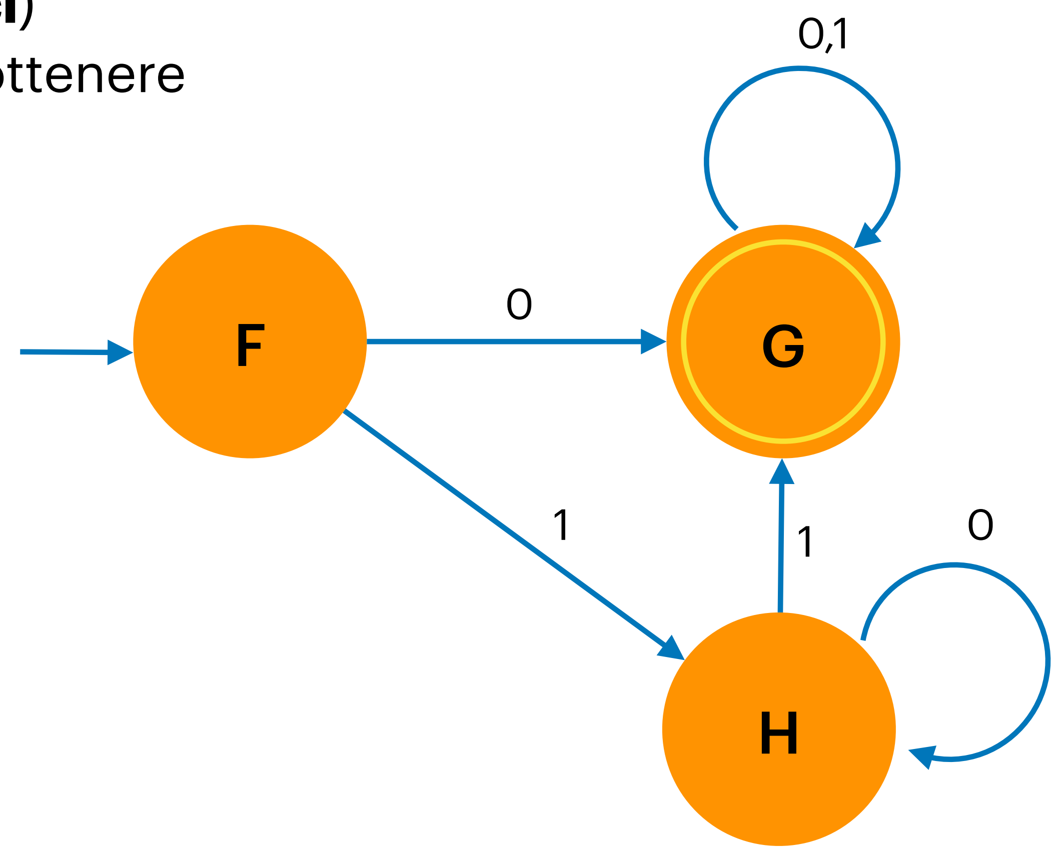


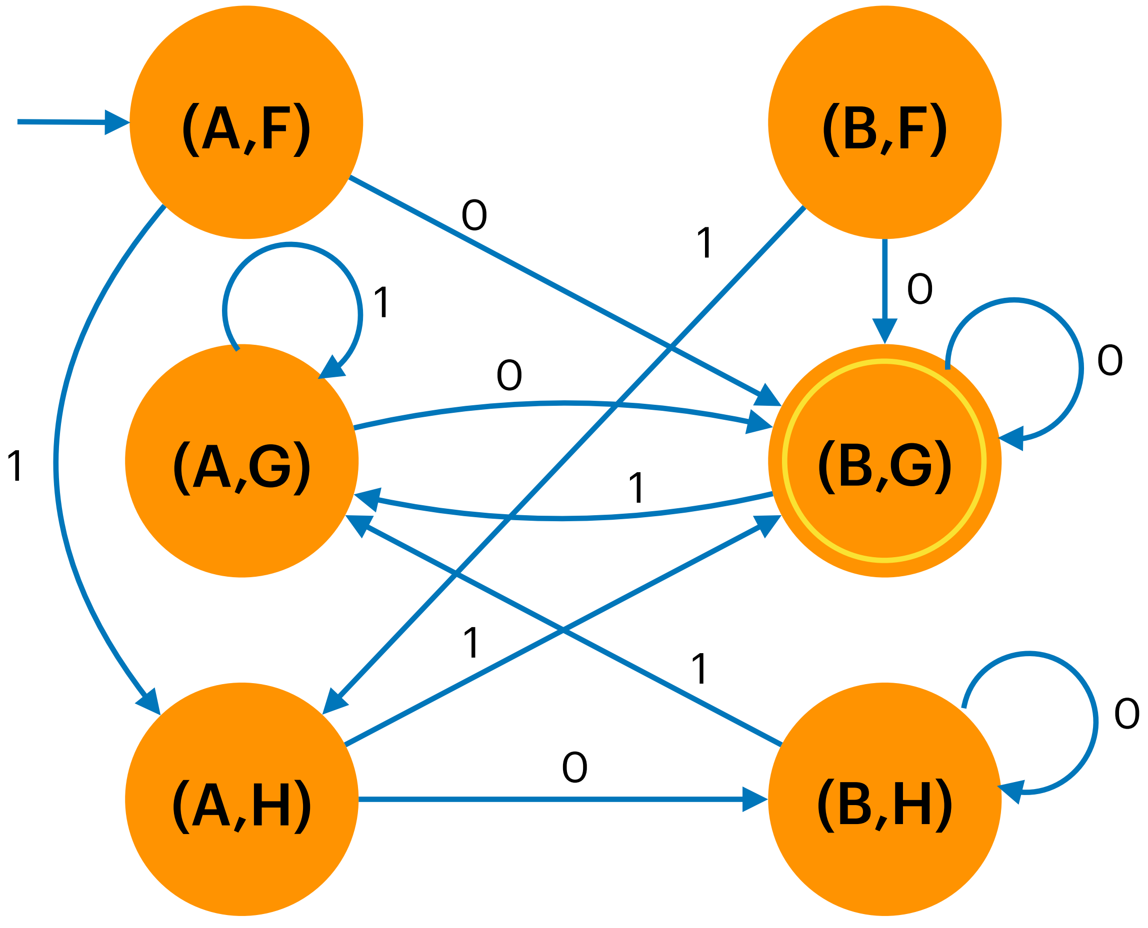


Prodotto di automi

E per gli stati finali? Se abbiamo DFA (automi **deterministici**) allora abbiamo diverse scelte a seconda di cosa vogliamo ottenere



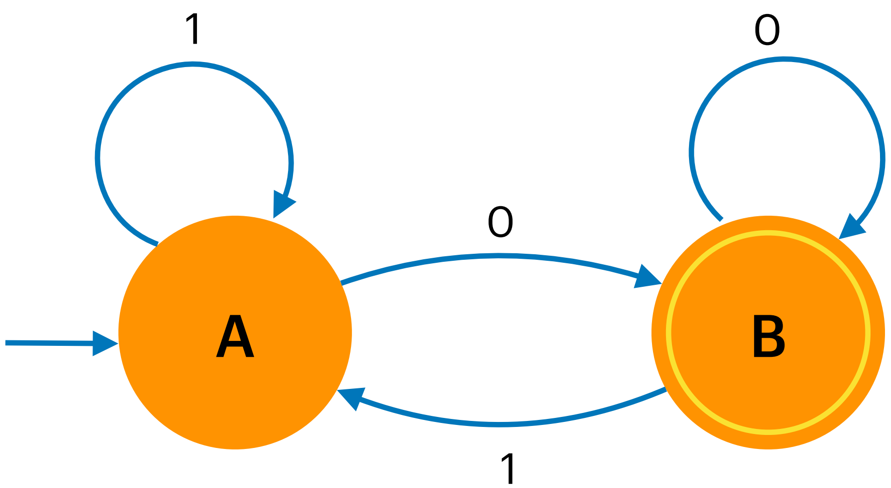


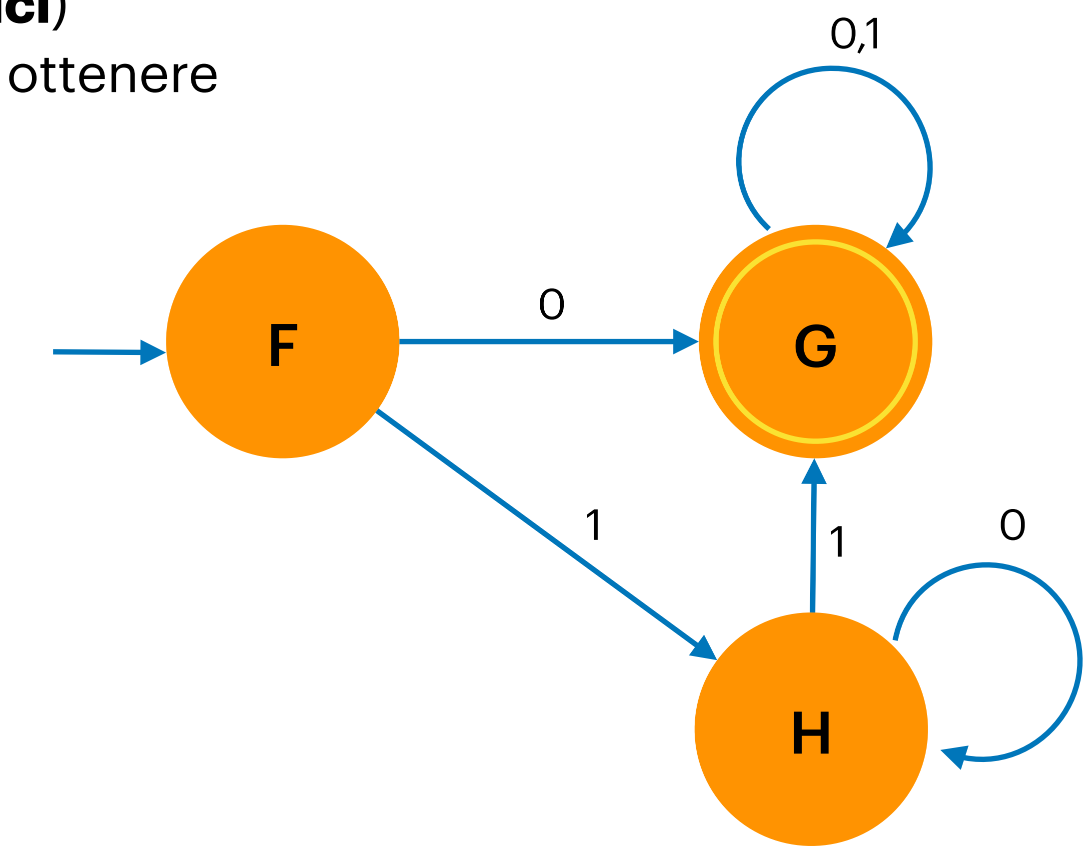


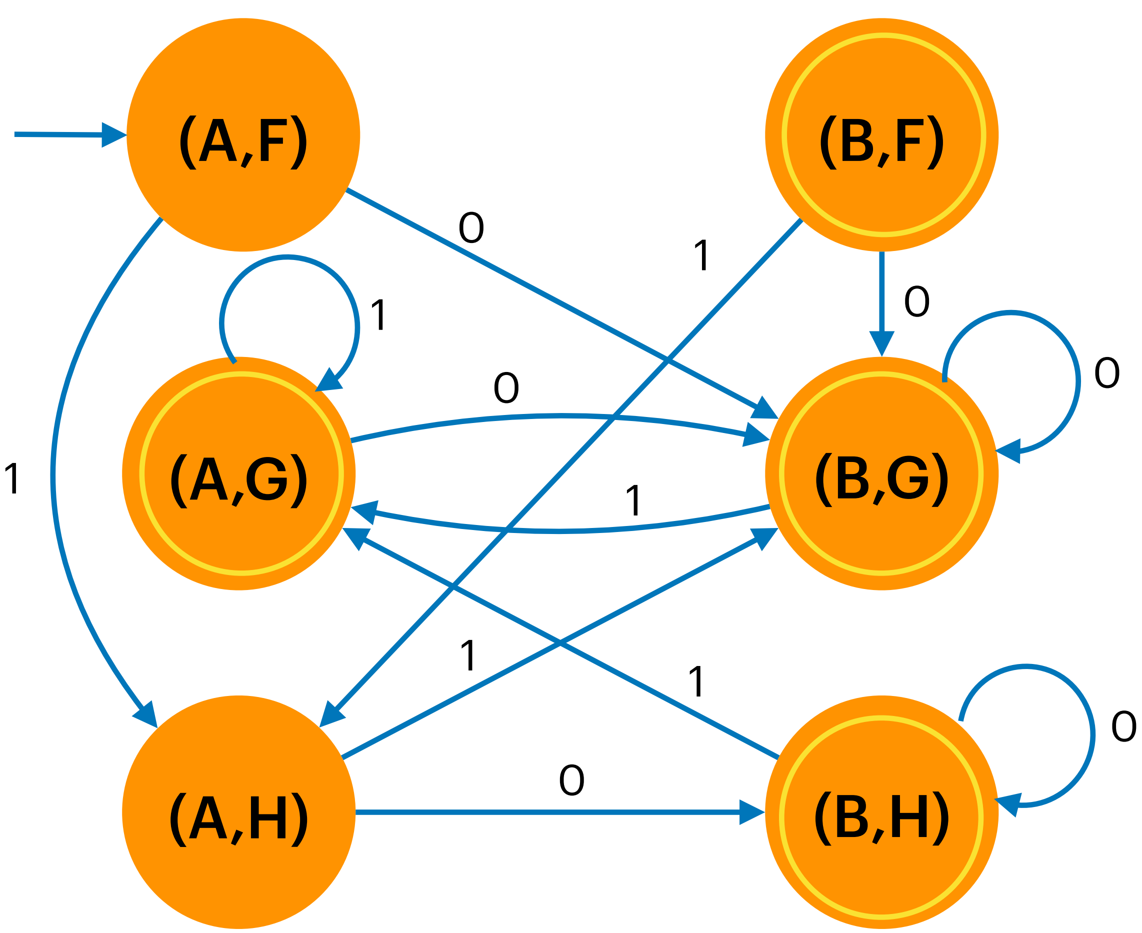
Se $F = \{(q, r) : q \in F_1 \text{ e } r \in F_2\}$ allora accettiamo $L_1 \cap L_2$

Prodotto di automi

E per gli stati finali? Se abbiamo DFA (automi **deterministici**) allora abbiamo diverse scelte a seconda di cosa vogliamo ottenere



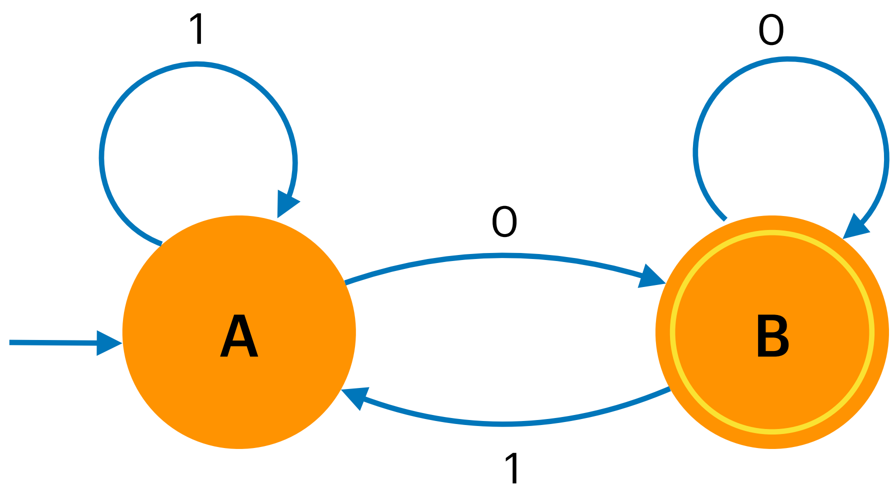


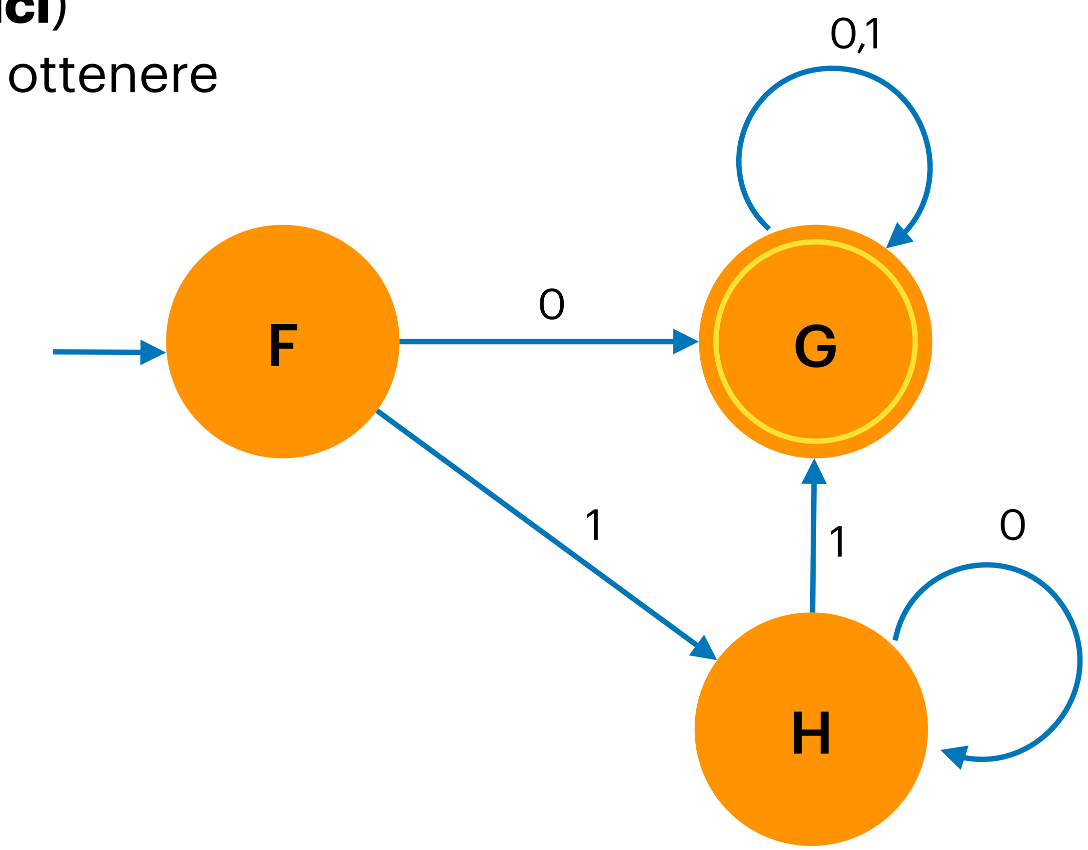


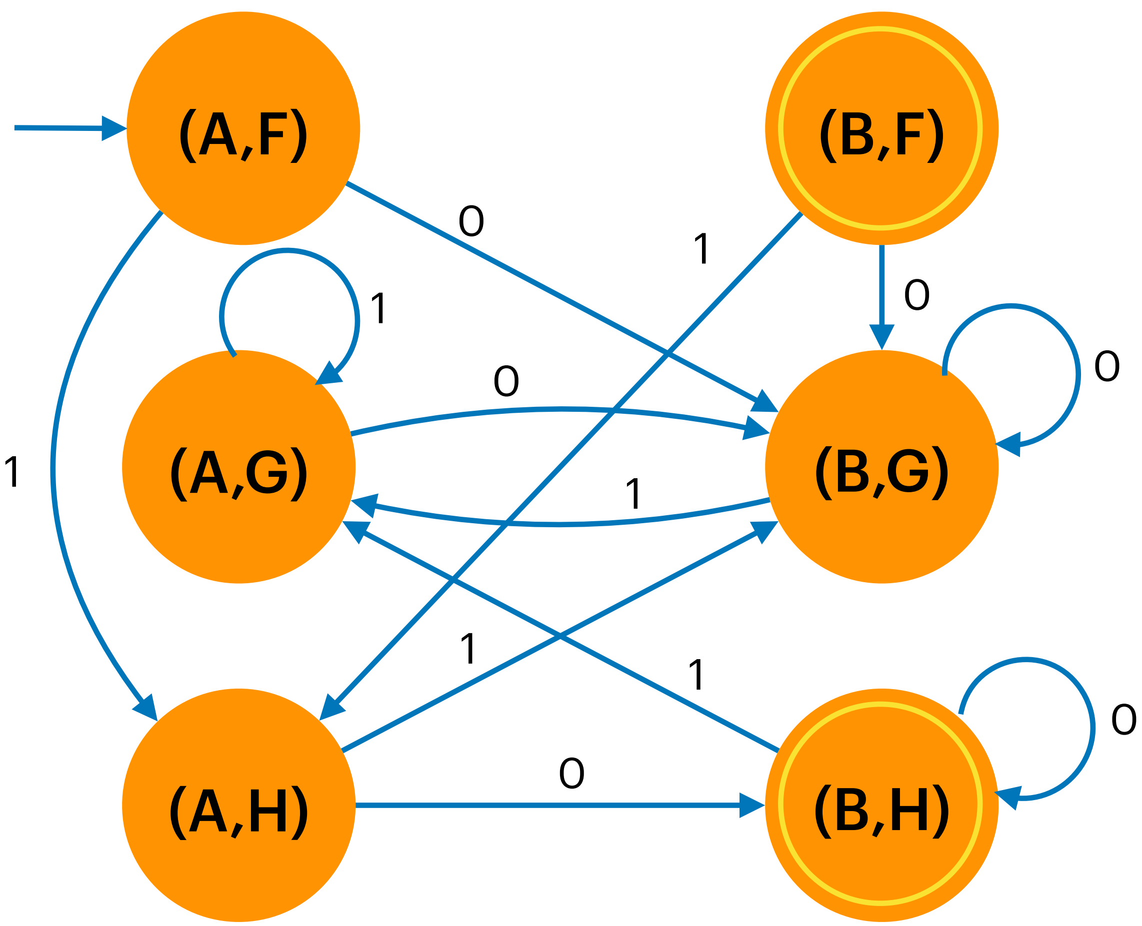
Se $F = \{(q, r) : q \in F_1$ oppure $r \in F_2\}$ allora accettiamo $L_1 \cup L_2$

Prodotto di automi

E per gli stati finali? Se abbiamo DFA (automi **deterministici**)
allora abbiamo diverse scelte a seconda di cosa vogliamo ottenere







Se $F = \{(q, r) : q \in F_1 \text{ e } r \notin F_2\}$
allora accettiamo $L_1 - L_2$

Prodotto di automi

- Grazie al prodotto di automi possiamo mostrare che i linguaggi regolari sono chiusi rispetto a:
    - Intersezione
    - Unione (costruzione alternativa all'uso di ε-transizioni)
    - Differenza
- Possiamo però trovare linguaggi che **non** sono regolari?
- Sì, usando il **pumping lemma per linguaggi regolari**

Pumping Lemma per linguaggi regolari

Per ogni linguaggio regolare $L \subseteq \Sigma^\star$ esiste una costante $n \in \mathbb{N}$ (dipendente dal linguaggio) tale per cui

ogni $w \in L$ con $|w| \geq n$ può essere scomposta come $w = xyz$ con:

1. $y \neq \varepsilon$ La stringa y non è la stringa vuota
2. $|xy| \leq n$ xy è una stringa lunga al più n
3. Per ogni $k \in \mathbb{N}$ la parola xy^kz è in L

Possiamo generare altre parole del linguaggio ripetendo 0 o più volte la stringa centrale (y)

Pumping Lemma per linguaggi regolari

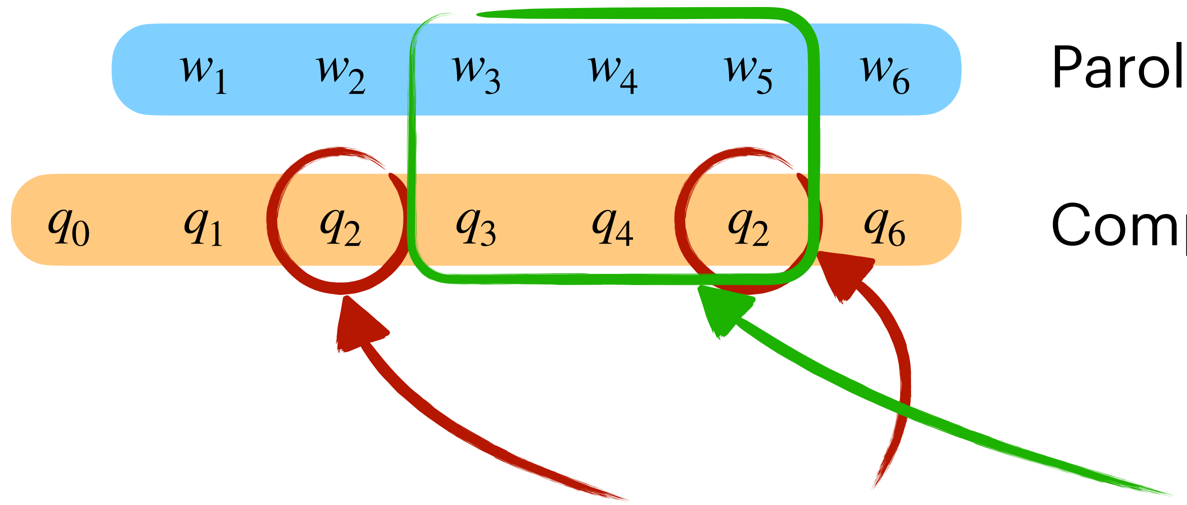
Idea della dimostrazione:

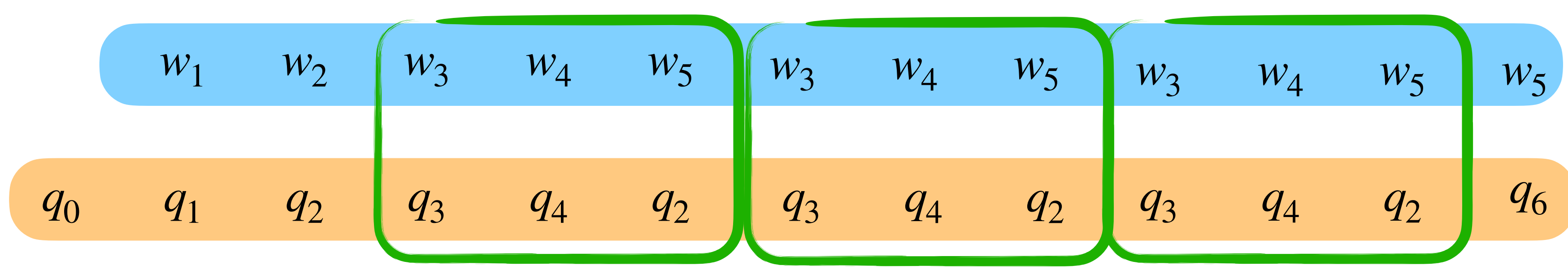
Supponiamo di riconoscere L con un automa di n stati

Se la parola da riconoscere $w \in L$ è lunga almeno n simboli allora una computazione che la riconosce passa almeno due volte per lo stesso stato q

Possiamo fare "copia e incolla" del pezzo di computazione tra quando appare q la prima volta e quando appare la seconda volta e ottenere ancora computazioni (valide) accettanti

Pumping Lemma per linguaggi regolari





Anche questa computazione è accettante e quindi
la parola è parte del linguaggio

In particolare abbiamo $x = w_1w_2$, $y = w_3w_4w_5$ e $z = w_6$

Pumping Lemma per linguaggi regolari

Per cosa serve il pumping lemma?

Ci aiuta a trovare linguaggi non regolari:

Se per ogni lunghezza n esiste una parola $w \in L$ di lunghezza almeno n tale per cui in qualsiasi modo si spezzi in $w = xyz$ non è vero che $xy^kz \in L$ per ogni $k \in \mathbb{N}$...

...allora L **non** è regolare

Esempi:
$L_1 = \{a^n b^n \; n \in \mathbb{N}\}$
L_2 il linguaggio delle parentesi bilanciate