Espressioni regolari

In aritmetica possiamo usare le operazioni $+$ e $\times$ per costruire operazioni del tipo

$$(6 + 3) \times 8.$$

Analogamente, possiamo utilizzare le **operazioni regolari** (unione, concatenazione e star) per costruire espressioni (chiamate **espressioni regolari**) del tipo

$$(0 \cup 1)0^*.$$

Il valore dell'espressione aritmetica è 72. Il valore dell'espressione regolare è un **linguaggio**.

Abbiamo visto che un **automa a stati finiti** (DFA o NFA) è un modo per definire **linguaggi regolari**.

Vedremo che una **espressione regolare** è un modo dichiarativo per descrivere un **linguaggio regolare**.

Espressioni regolari

Consideriamo l'alfabeto $\Sigma = \{0, 1\}$. Nelle espressioni regolari per brevità

- 0 indica l'insieme $\{0\}$;
- 1 indica l'insieme $\{1\}$.

Quindi l'espressione $0 \cup 1$ indica $\{0\} \cup \{1\}$, cioè $\{0, 1\}$

L'espressione $(0 \cup 1)0^*$ corrisponde a $\{0, 1\} \circ \{0\}^*$. Questo è l'insieme delle stringhe binarie che iniziano con 0 oppure 1 e continuano con degli 0 (anche nessuno).

L'espressione $(0 \cup 1)^*$ corrisponde a $\{0, 1\}^*$, cioè l'insieme di tutte le stringhe binarie.

Vediamo ora di dare una definizione formale di **espressione regolare**.

Espressioni regolari: definizione induttiva

Base.

- a è un'espressione regolare per ogni $a \in \Sigma$;
- ϵ è un'espressione regolare (denota $\{\epsilon\}$);
- $\emptyset$ è un'espressione regolare;

Passo.

- se R_1 e R_2 sono espressioni regolari, allora $R_1 \cup R_2$ è un'espressione regolare;
- se R_1 e R_2 sono espressioni regolari, allora $R_1 \circ R_2$ è un'espressione regolare;
- se R è un'espressione regolare, allora R^* è un'espressione regolare.

Dimostrare proprietà sulle espressioni regolari

La definizione induttiva suggerisce un metodo per dimostrare che una espressione regolare R soddisfa una certa proprietà.

(**Base**) Si dimostra che la proprietà è soddisfatta per i casi base:

- è soddisfatta per $R = a$, dove $a \in \Sigma$;
- è soddisfatta per $R = \epsilon$;
- è soddisfatta per $R = \emptyset$.

(**Passo induttivo**) Si suppone per ipotesi induttiva che la proprietà sia soddisfatta da R_1 e R_2 e si dimostra che è soddisfatta anche da ciascuna delle seguenti espressioni regolari:

- $R = R_1 \cup R_2$;
- $R = R_1 \circ R_2$;
- $R = R_1^*$.

Ordine di precedenza delle operazioni regolari

In aritmetica, la moltiplicazione ha precedenza sull'addizione.

$$2 + 3 \times 4 = 14.$$

Le parentesi possono essere usate per cambiare l'ordine usuale:

$$(2 + 3) \times 4 = 20.$$

Le operazioni regolari hanno il seguente **ordine di precedenza**:

1. Kleene star.
2. Concatenazione.
3. Unione.

Le parentesi cambiano l'ordine usuale.

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

$00 \cup 101^*$ descrive ...

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

$00 \cup 101^*$ descrive ...
il linguaggio formato dalla stringa 00 e da ogni stringa iniziante con 10 e seguita da zero o più 1.

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

$00 \cup 101^*$ descrive ...
il linguaggio formato dalla stringa 00 e da ogni stringa iniziante con 10 e seguita da zero o più 1.

$0(0 \cup 101)^*$

- contiene 0101001010?

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

$00 \cup 101^*$ descrive ...
il linguaggio formato dalla stringa 00 e da ogni stringa iniziante con 10 e seguita da zero o più 1.

$0(0 \cup 101)^*$

- contiene 0101001010?
- contiene 00101001?

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

$00 \cup 101^*$ descrive ...
il linguaggio formato dalla stringa 00 e da ogni stringa iniziante con 10 e seguita da zero o più 1.

$0(0 \cup 101)^*$

- contiene 0101001010?
- contiene 00101001?
- contiene 0000000?

Esempi

$01^* \cup 1$ corrisponde a $(0(1)^*) \cup 1$ e descrive ...
il linguaggio formato dalla stringa 1 e da ogni stringa iniziante con 0 e seguita da zero o più 1.

$00 \cup 101^*$ descrive ...
il linguaggio formato dalla stringa 00 e da ogni stringa iniziante con 10 e seguita da zero o più 1.

$0(0 \cup 101)^*$

- contiene 0101001010?
- contiene 00101001?
- contiene 0000000?
- contiene 101?

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. 0^*10^*

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w | \ w \text{ contiene un singolo } 1\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w | \ w \text{ contiene un singolo } 1\}$.
2. $\Sigma^*1\Sigma^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w | \ w \text{ contiene un singolo } 1\}$.
2. $\Sigma^*1\Sigma^* = \{w | \ w \text{ ha almeno un } 1\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w | \ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w | \ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |\$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |\ $ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w|\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w|\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w|\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w|$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w|\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w|$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w|\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w|\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w|\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w|\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w|$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w|\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w|$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w|\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.
10. $(0 \cup \epsilon)(1 \cup \epsilon)$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.
10. $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.
10. $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.
11. $1^*\emptyset$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.
10. $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.
10. $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
12. $\emptyset^*$

Esempi

Sia $\Sigma = \{0, 1\}$. Definiamo il linguaggio descritto da ciascuna delle seguenti espressioni regolari.

1. $0^*10^* = \{w |\ w$ contiene un singolo $1\}$.
2. $\Sigma^*1\Sigma^* = \{w |\ w$ ha almeno un $1\}$.
3. $\Sigma^*001\Sigma^* = \{w |\ w$ contiene la stringa 001 come sottostringa$\}$.
4. $1^*(01^+)^* = \{w |$ ogni 0 in w è seguito da almeno un $1\}$.
5. $(\Sigma\Sigma)^* = \{w |\ w$ è una stringa di lunghezza pari$\}$.
6. $(\Sigma\Sigma\Sigma)^* = \{w |$ la lunghezza di w è un multiplo di tre$\}$.
7. $01 \cup 10 = \{01, 10\}$.
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w |\ w$ comincia e finisce con lo stesso simbolo$\}$.
9. $(0 \cup \epsilon)1^* = 01^* \cup 1^*$.
10. $(0 \cup \epsilon)(1 \cup \epsilon) = \{\epsilon, 0, 1, 01\}$.
11. $1^*\emptyset = \emptyset$.
12. $\emptyset^* = \{\epsilon\}$.

Teorema di Kleene

Espressioni regolari e automi finiti sono equivalenti nella loro potenza descrittiva.

Ogni espressione regolare può essere convertita in un automa finito che riconosce il linguaggio che essa descrive e viceversa.
Data una espressione regolare R indichiamo con $L(R)$ il linguaggio descritto da R.

Teorema

Un linguaggio è regolare se e solo se esiste una espressione regolare che lo descrive.

Idea. Sappiamo che un linguaggio L è regolare **se e solo se** L è riconosciuto da un NFA **se e solo se** L è riconosciuto da un DFA

Dimostriamo che

- per ogni espressione regolare R esiste un NFA N, tale che $L(N) = L(R)$;
- per ogni DFA M possiamo costruire un'espressione regolare R con $L(R) = L(M)$.

Cioè L è riconosciuto da un DFA **se e solo se** L può essere descritto da un'espressione regolare.

Teorema di Kleene

Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

$R = a$ per qualche $a \in \Sigma$. Allora $L(R) = \{a\}$. Costruiamo un NFA che riconosca $L(R)$:

Teorema di Kleene

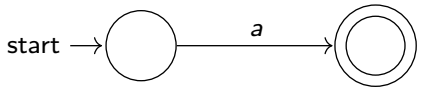
Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

$R = a$ per qualche $a \in \Sigma$. Allora $L(R) = \{a\}$. Costruiamo un NFA che riconosca $L(R)$:



Teorema di Kleene

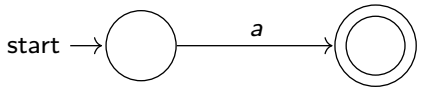
Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

$R = a$ per qualche $a \in \Sigma$. Allora $L(R) = \{a\}$. Costruiamo un NFA che riconosca $L(R)$:



$R = \epsilon$. Allora $L(R) = \{\epsilon\}$. Costruiamo un NFA che riconosca $L(R)$:

Teorema di Kleene

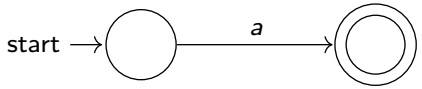
Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

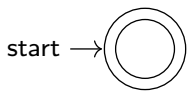
Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

$R = a$ per qualche $a \in \Sigma$. Allora $L(R) = \{a\}$. Costruiamo un NFA che riconosca $L(R)$:



$R = \epsilon$. Allora $L(R) = \{\epsilon\}$. Costruiamo un NFA che riconosca $L(R)$:



Teorema di Kleene

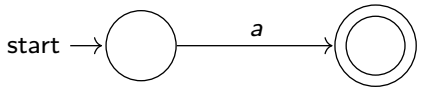
Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

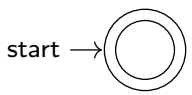
Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

$R = a$ per qualche $a \in \Sigma$. Allora $L(R) = \{a\}$. Costruiamo un NFA che riconosca $L(R)$:



$R = \epsilon$. Allora $L(R) = \{\epsilon\}$. Costruiamo un NFA che riconosca $L(R)$:



$R = \emptyset$. Allora $L(R) = \emptyset$. Costruiamo un NFA che riconosca $L(R)$:

Teorema di Kleene

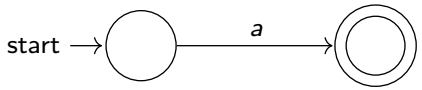
Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

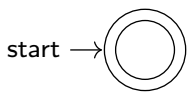
Dimostrazione.
Supponiamo di avere un'espressione regolare R che descrive un linguaggio A.
Trasformiamo R in un NFA che riconosce A. Questo implica che A è regolare.

Casi base.

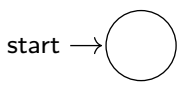
$R = a$ per qualche $a \in \Sigma$. Allora $L(R) = \{a\}$. Costruiamo un NFA che riconosca $L(R)$:



$R = \epsilon$. Allora $L(R) = \{\epsilon\}$. Costruiamo un NFA che riconosca $L(R)$:



$R = \emptyset$. Allora $L(R) = \emptyset$. Costruiamo un NFA che riconosca $L(R)$:



Lemma

Se un linguaggio è descritto da un'espressione regolare, allora esso è regolare.

Dimostrazione.

Passo induttivo.

Assumiamo per ipotesi induttiva che il lemma sia valido per R_1 e R_2, cioè che esistano NFA che riconoscono $L(R_1)$ e $L(R_2)$. Dobbiamo provare che è valido anche per $R_1 \cup R_2$, $R_1 R_2$ e R_1^*.

Immediato perché abbiamo già visto come costruire un NFA per $L(R_1 \cup R_2)$, $L(R_1 R_2)$ e $L(R_1^*)$: i linguaggi regolari sono chiusi rispetto alle tre operazioni regolari.

Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

Esempio 1

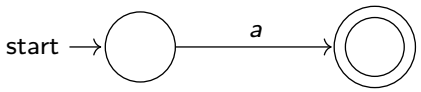
Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

a

Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

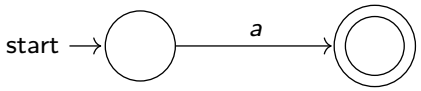
a



Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

a

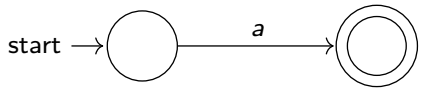


b

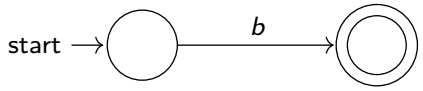
Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

a



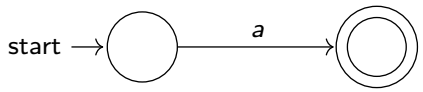
b



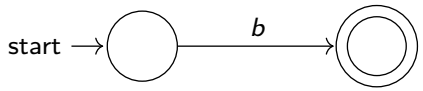
Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

a



b

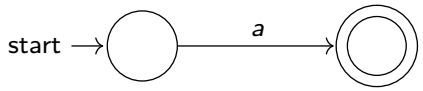


ab

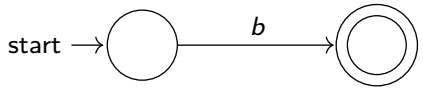
Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

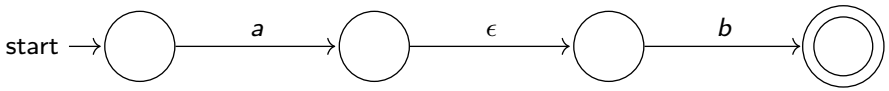
a



b



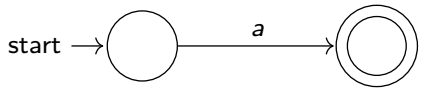
ab



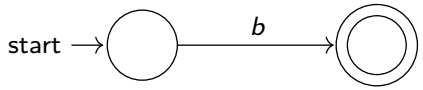
Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

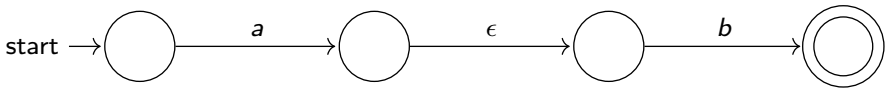
a



b



ab

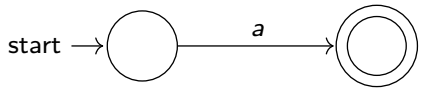


$ab \cup a$

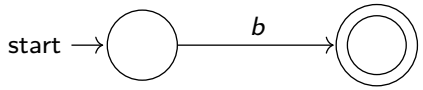
Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

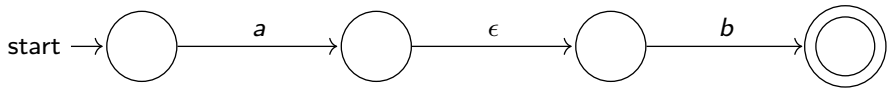
a



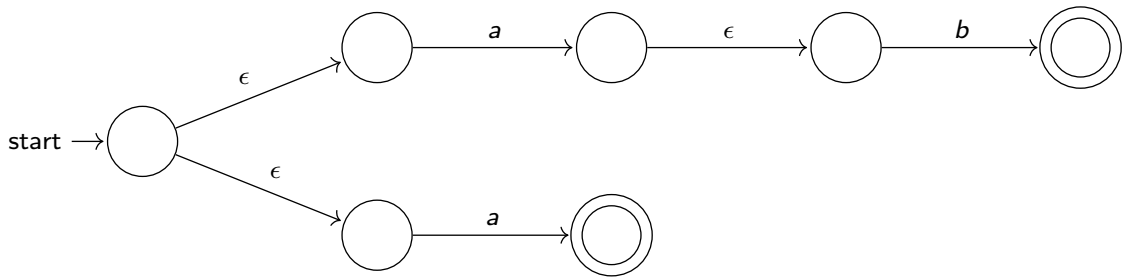
b



ab



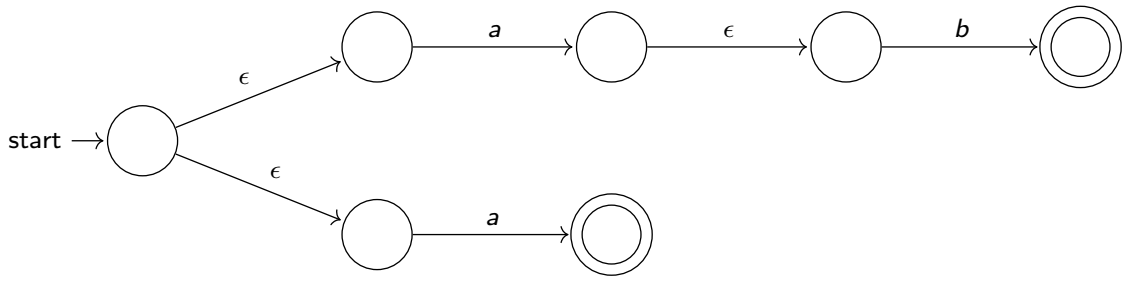
$ab \cup a$



Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

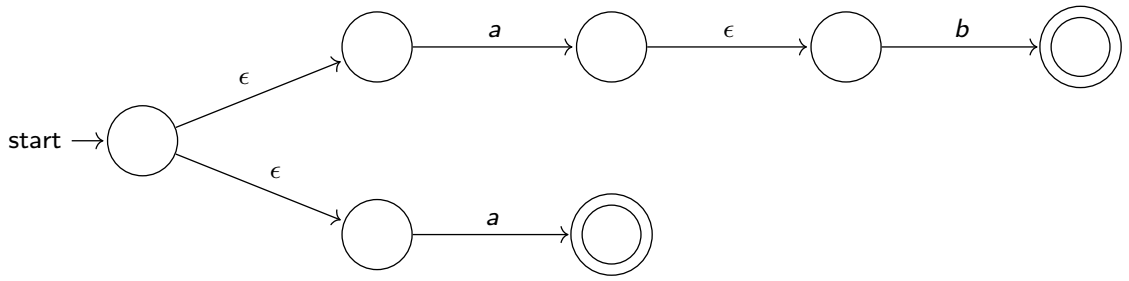
$ab \cup a$



Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

$ab \cup a$

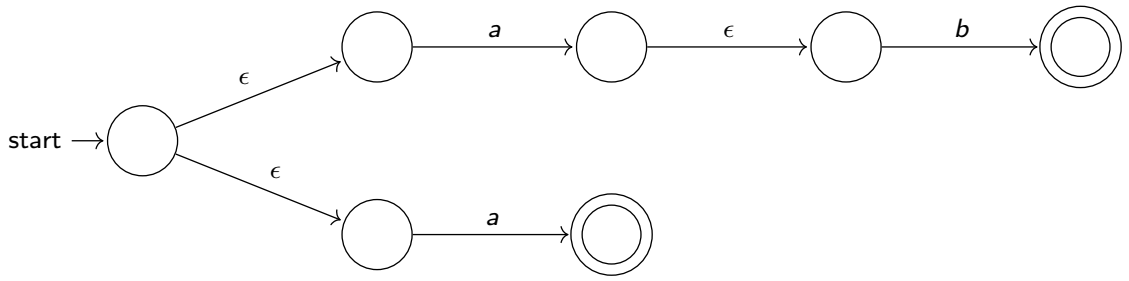


$(ab \cup a)^*$

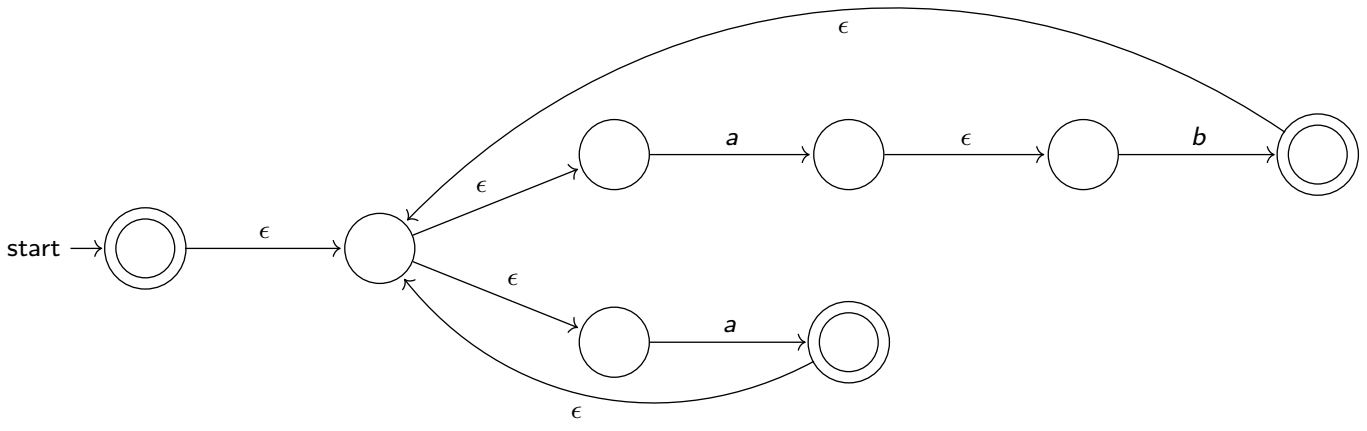
Esempio 1

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(ab \cup a)^*$.

$ab \cup a$



$(ab \cup a)^*$



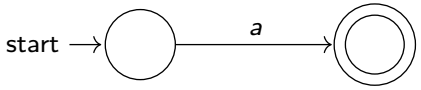
Esempio 2

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(a \cup b)^* aba$.

Esempio 2

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(a \cup b)^* aba$.

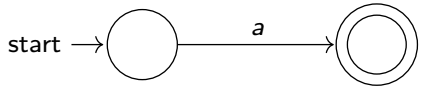
a



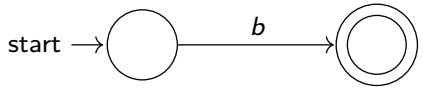
Esempio 2

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(a \cup b)^* aba$.

a



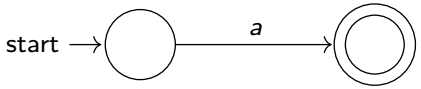
b



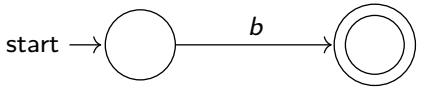
Esempio 2

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(a \cup b)^* aba$.

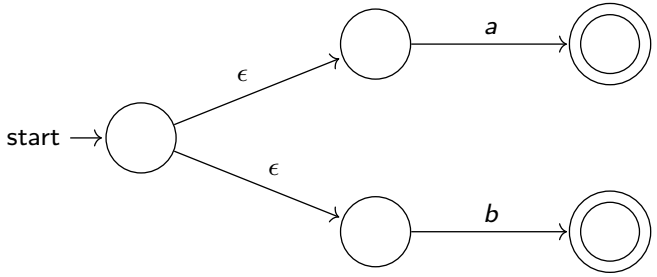
a



b



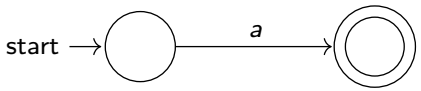
$a \cup b$



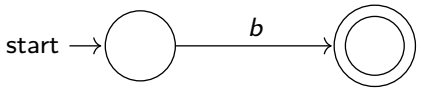
Esempio 2

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(a \cup b)^* aba$.

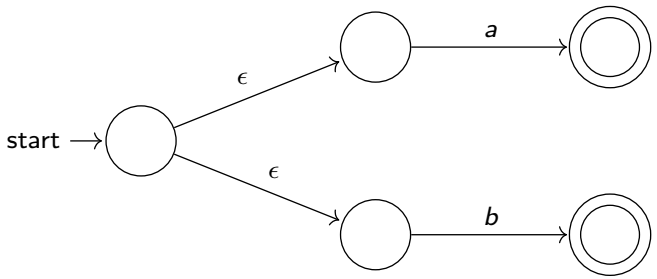
a



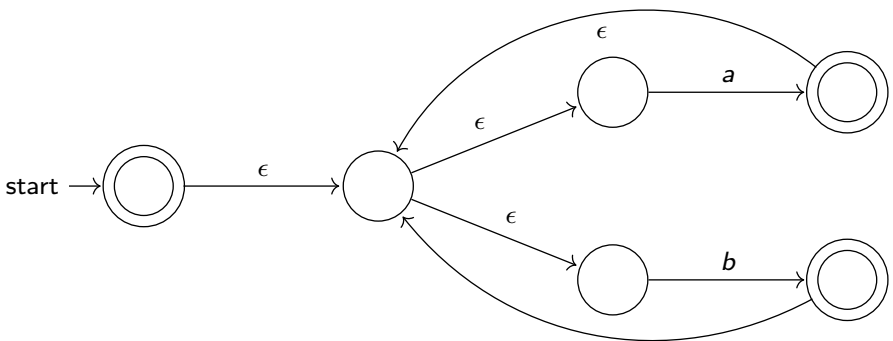
b



$a \cup b$



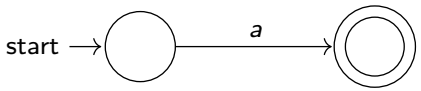
$(a \cup b)^*$



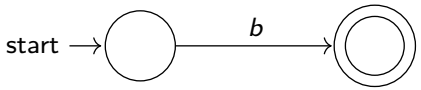
Esempio 2

Troviamo un NFA che riconosca il linguaggio descritto dall'espressione regolare $(a \cup b)^* aba$.

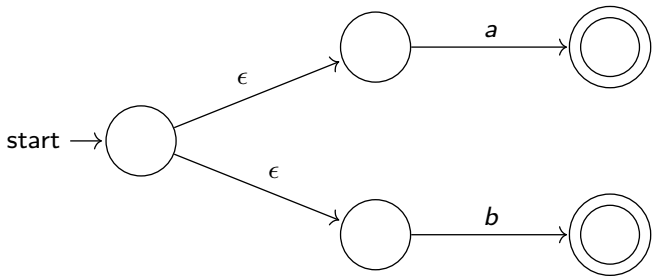
a



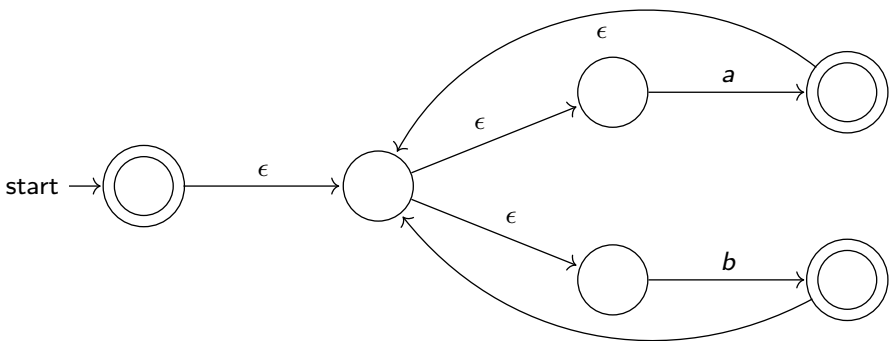
b



$a \cup b$

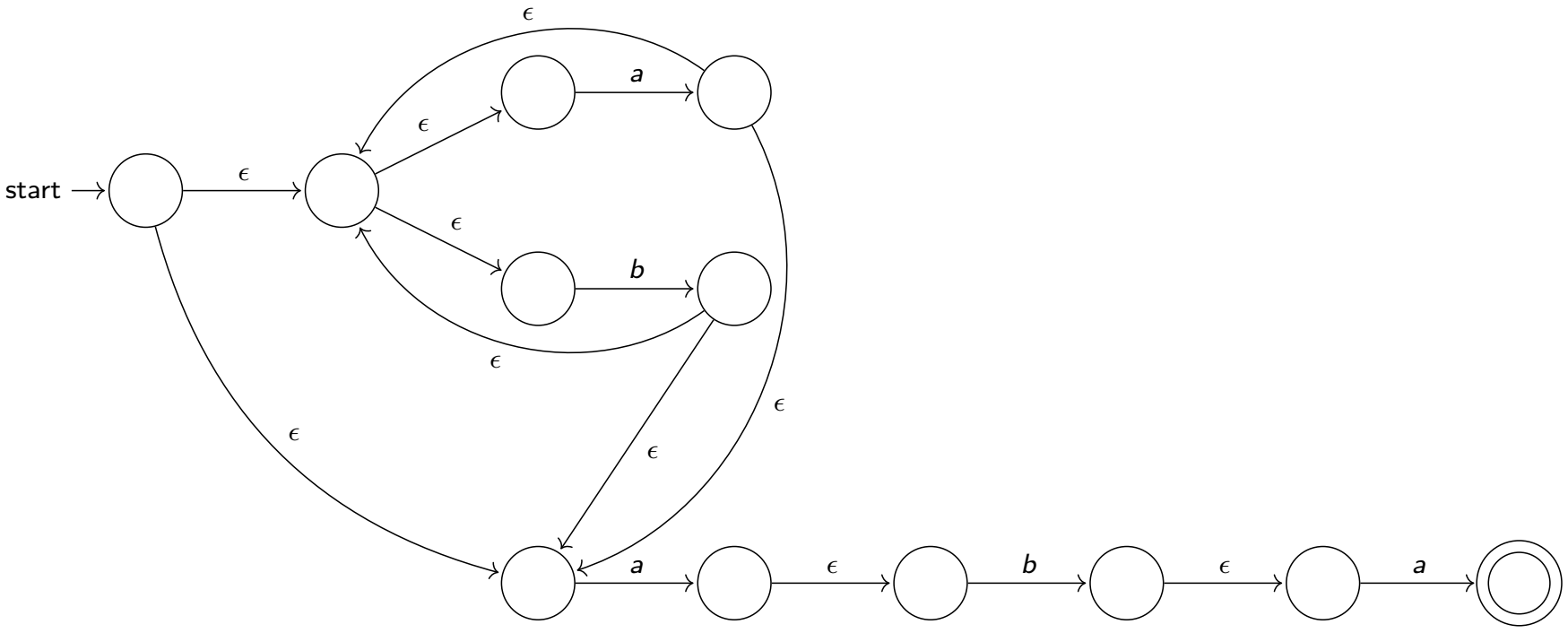


$(a \cup b)^*$



Esempio 2

$(a \cup b)^* aba$

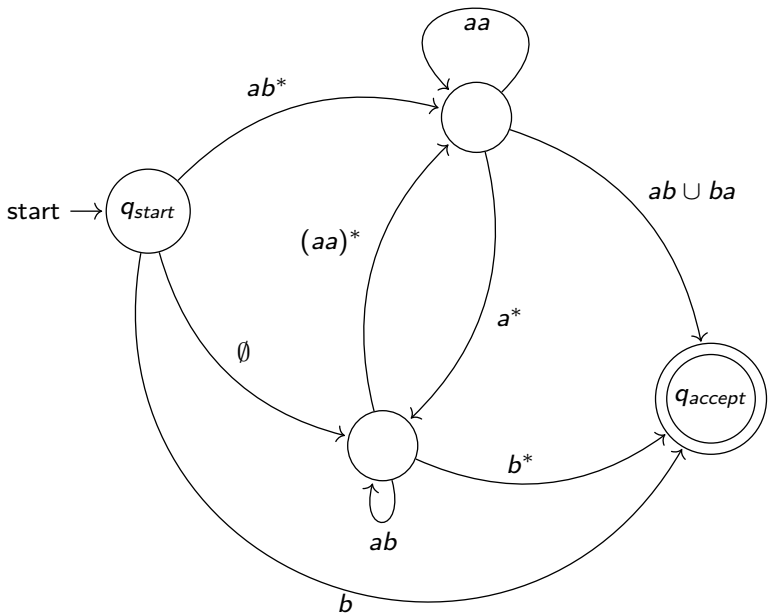


Teorema di Kleene

Lemma

Se un linguaggio è regolare, allora è descritto da un'espressione regolare.

Dimostrazione.
Descriviamo una procedura per trasformare i DFA in espressioni regolari equivalenti.
Useremo una generalizzazione dell'NFA: **automa finito non deterministico generalizzato (GNFA)** che è un NFA che permette archi etichettati con espressioni regolari.



Dimostreremo prima come convertire DFA in GNFA e poi GNFA in espressioni regolari.

Teorema di Kleene

Useremo GNFA (forma speciale) che soddisfano i seguenti requisiti:

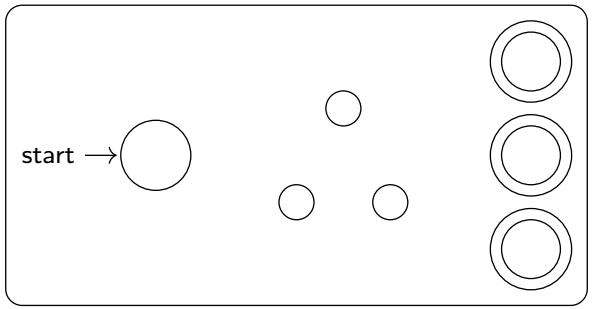
1) Lo stato iniziale ha archi verso ogni altro stato ma non ha archi entranti.

2) Esiste un unico stato accettante che può avere archi provenienti da qualsiasi altro stato, ma non ha archi uscenti. Inoltre stato accettante e stato iniziale sono diversi.

Teorema di Kleene

Convertiamo un DFA in un GNFA in forma speciale.

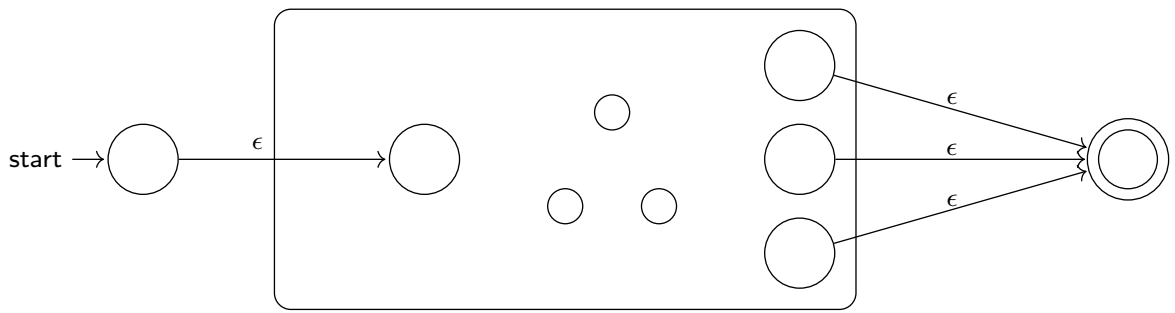
1. Aggiungiamo un nuovo stato iniziale con un ϵ-arco verso il vecchio stato iniziale.
2. Aggiungiamo un nuovo stato accettante con ϵ-archi entranti provenienti dai vecchi stati accettanti.



Teorema di Kleene

Convertiamo un DFA in un GNFA in forma speciale.

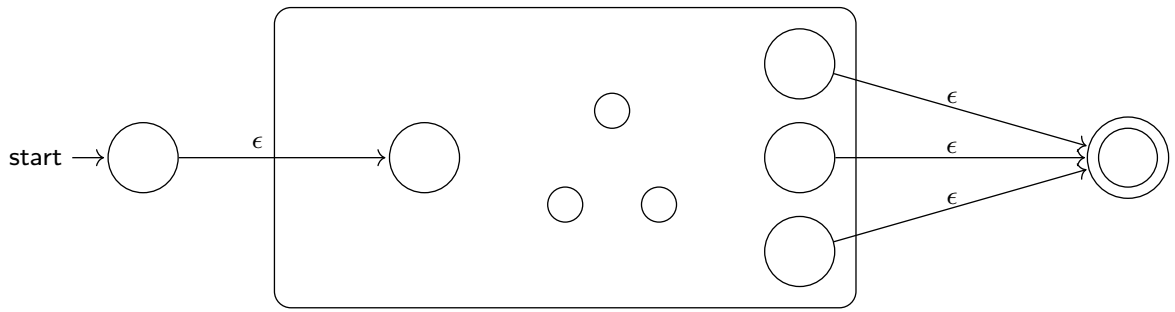
1. Aggiungiamo un nuovo stato iniziale con un ϵ-arco verso il vecchio stato iniziale.
2. Aggiungiamo un nuovo stato accettante con ϵ-archi entranti provenienti dai vecchi stati accettanti.



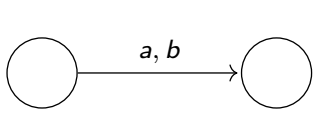
Teorema di Kleene

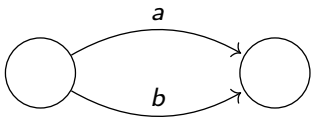
Convertiamo un DFA in un GNFA in forma speciale.

1. Aggiungiamo un nuovo stato iniziale con un ϵ-arco verso il vecchio stato iniziale.
2. Aggiungiamo un nuovo stato accettante con ϵ-archi entranti provenienti dai vecchi stati accettanti.



Eventuali archi con più etichette o archi che collegano 2 stati nella stessa direzione vengono sostituiti con un solo arco etichettato con l'unione delle precedenti etichette.

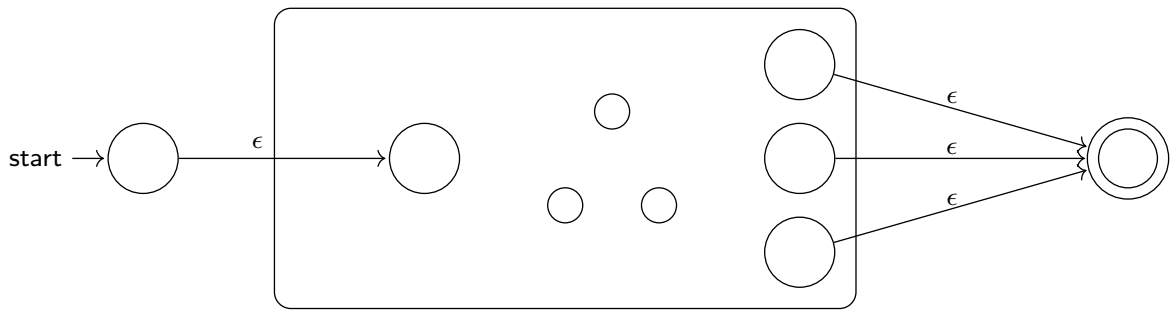




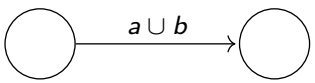
Teorema di Kleene

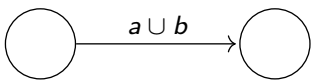
Convertiamo un DFA in un GNFA in forma speciale.

1. Aggiungiamo un nuovo stato iniziale con un ϵ-arco verso il vecchio stato iniziale.
2. Aggiungiamo un nuovo stato accettante con ϵ-archi entranti provenienti dai vecchi stati accettanti.



Eventuali archi con più etichette o archi che collegano 2 stati nella stessa direzione vengono sostituiti con un solo arco etichettato con l'unione delle precedenti etichette.



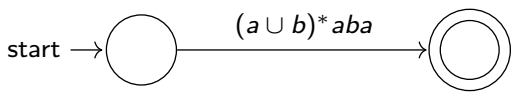


Teorema di Kleene

Abbiamo così ottenuto un GNFA in forma speciale equivalente al DFA di partenza. Trasformiamo ora questo GNFA in un'espressione regolare equivalente.

Sia k il numero di stati del GNFA. Chiaramente $k \geq 2$, visto che stato iniziale e stato accettante sono diversi.

Il nostro obiettivo è di convertire questo GNFA con $k \geq 2$ in un **GNFA equivalente con 2 stati**.

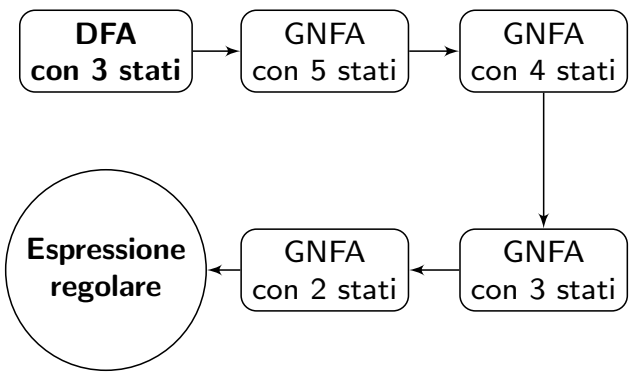


Per $k = 2$ il GNFA ha un solo arco dallo stato iniziale allo stato finale. L'etichetta di questo arco è l'espressione regolare equivalente (e la conversione è terminata).

Conversione da $k \geq 2$ stati a $k = 2$ stati.

Se $k > 2$ costruiamo un GNFA equivalente con $k - 1$ stati. Iteriamo la procedura fino ad arrivare ad un GNFA con $k = 2$ stati.

I passi della procedura di conversione partendo da un DFA con 3 stati:

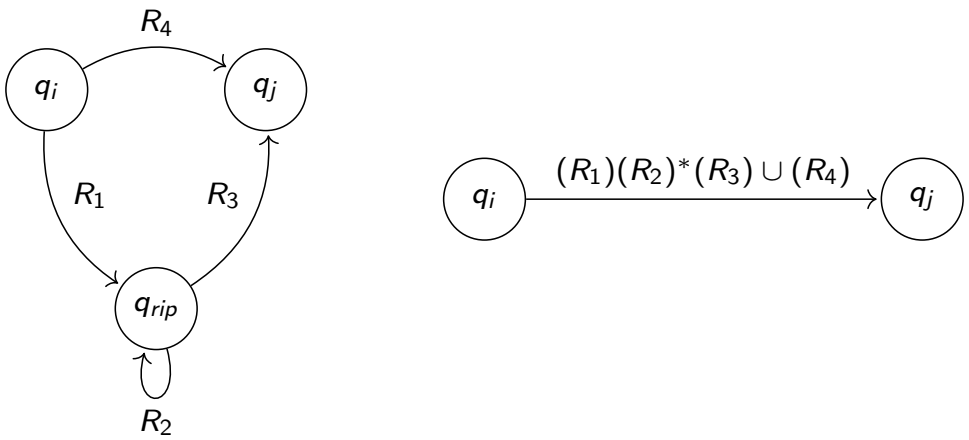


Conversione da $k > 2$ stati a $k - 1$ stati.

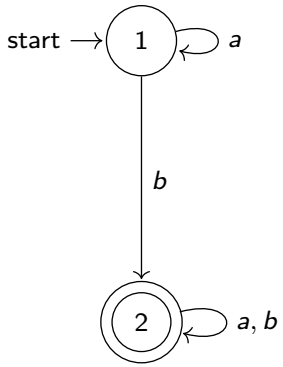
Scegliamo uno stato quasiasi che sia diverso da quello iniziale e da quello accettante e lo togliamo dalla macchina modificando tutto il resto in modo che la nuova macchina riconosca lo stesso linguaggio.

Sia q_{rip} lo stato rimosso. La macchina viene modificata nel modo seguente. Si cambiano le espressioni regolari che etichettano i restanti archi in modo da controbilanciare l'assenza di q_{rip} reintegrando le computazioni rimosse.

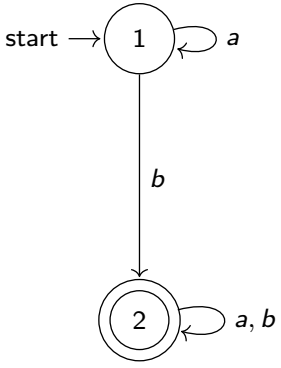
Ogni nuova etichetta che va da uno stato q_i a uno stato q_j è un'espressione regolare che descrive tutte le stringhe che avrebbero portato la macchina da q_i a q_j direttamente o attraverso $q_r ip$.

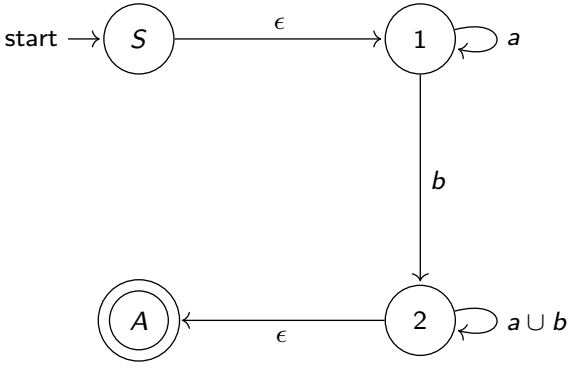


Esempio 1

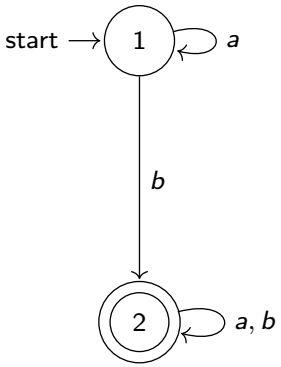


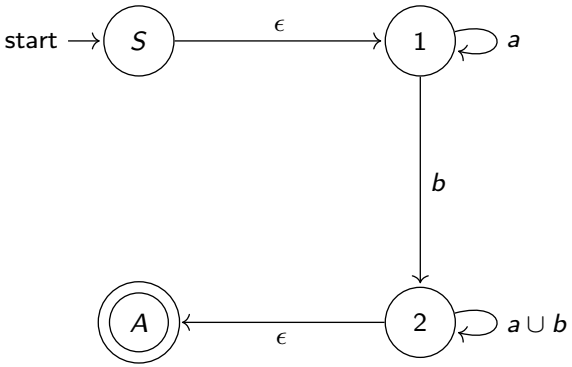
Esempio 1

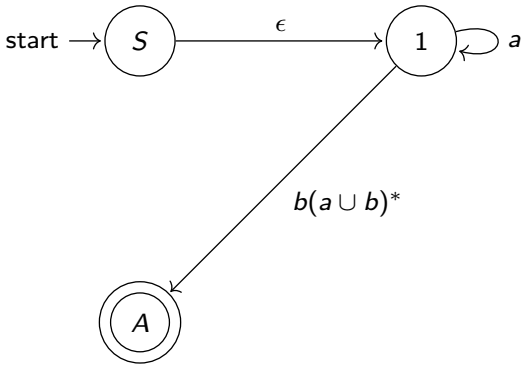




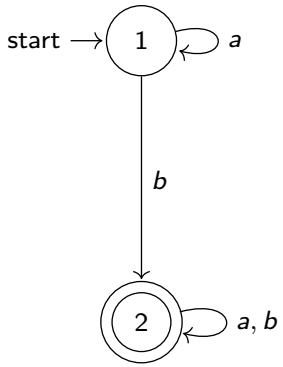
Esempio 1

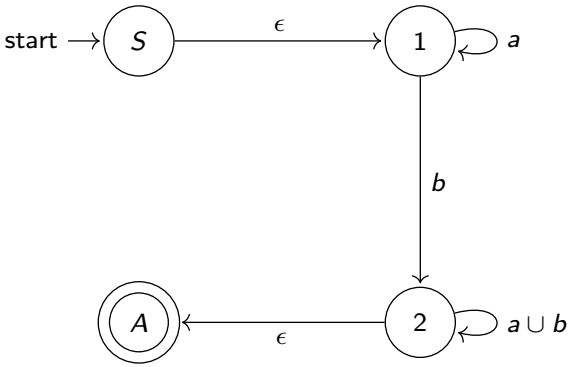


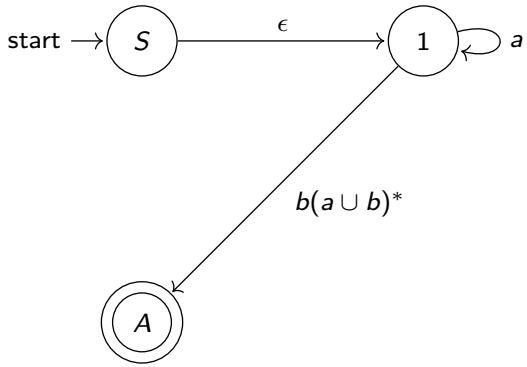


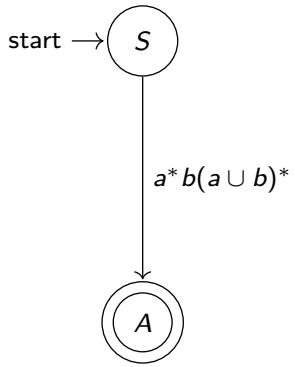


Esempio 1

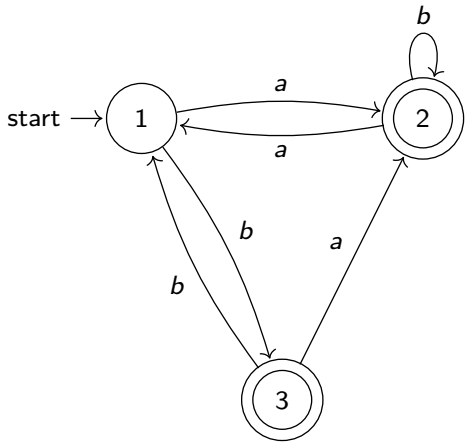

start
1
a
b
2
a, b


start
S
ϵ
1
a
b
A
ϵ
2
a ∪ b

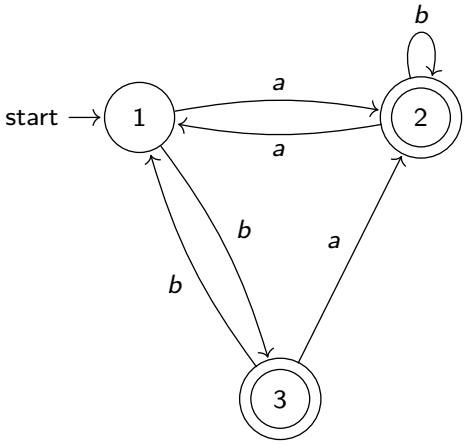

start
S
ϵ
1
a
b(a ∪ b)*
A

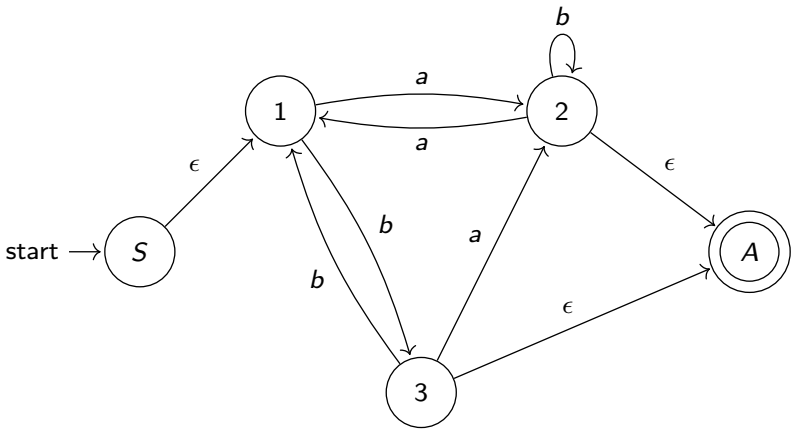

start
S
a*b(a ∪ b)*
A

Esempio 2

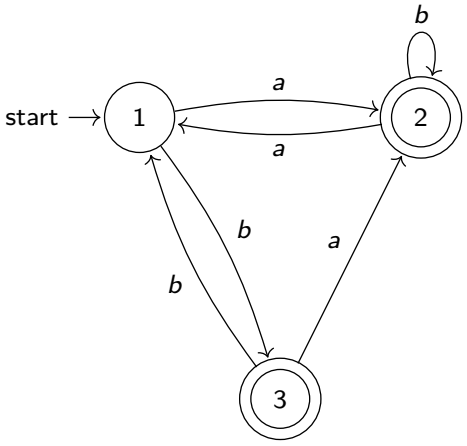


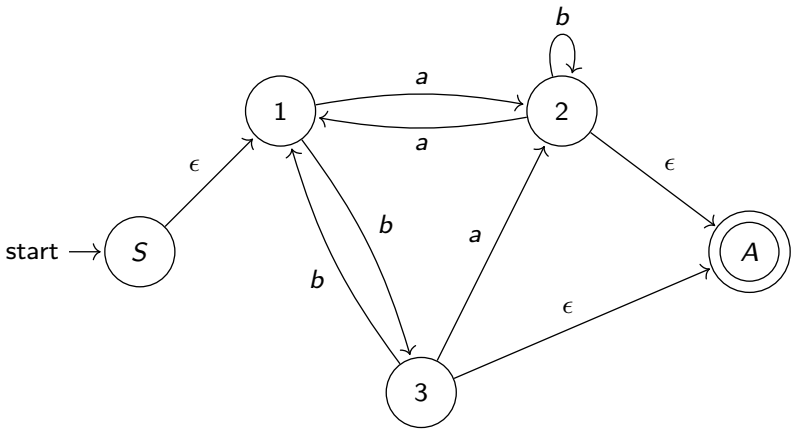
Esempio 2

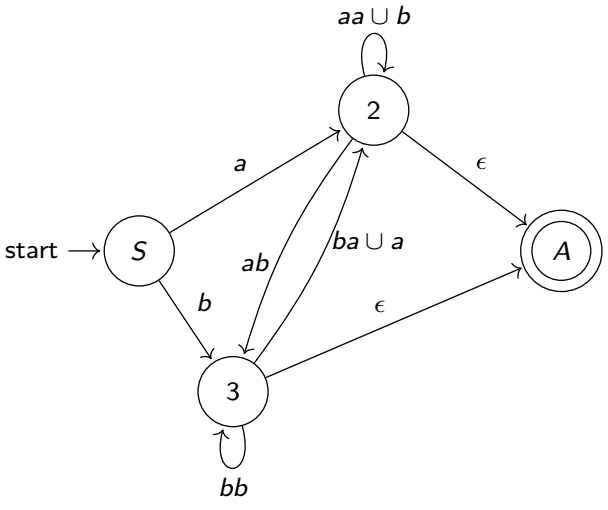




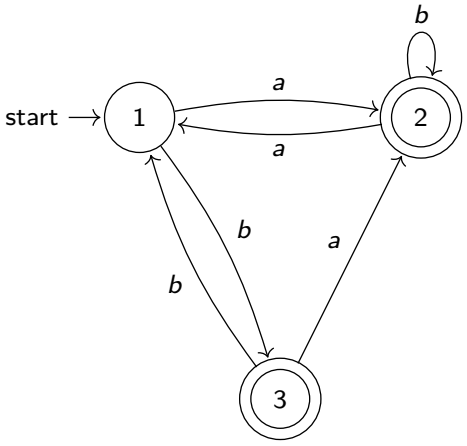
Esempio 2

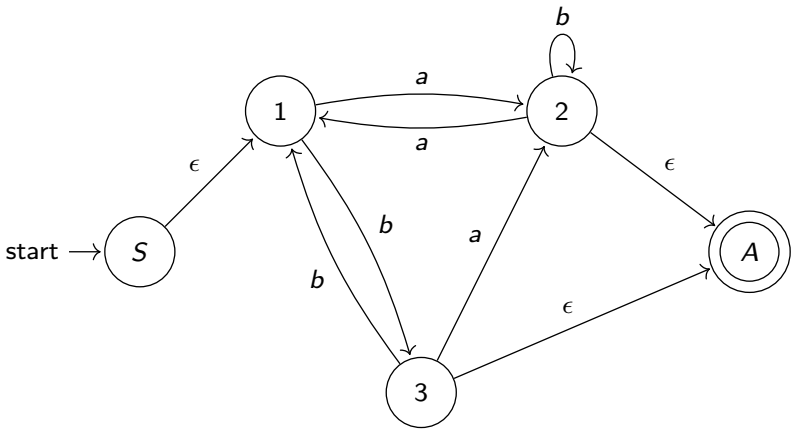

start
1
2
3
a
a
b
b
a
b

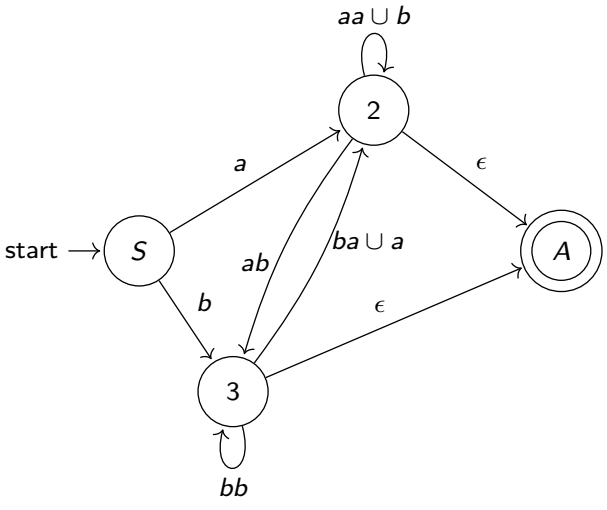

start
S
1
2
3
A
ϵ
a
a
b
b
a
ϵ
ϵ

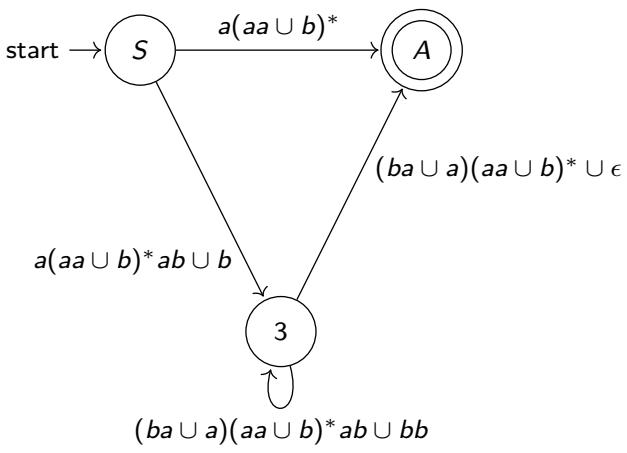

start
S
2
3
A
aa ∪ b
a
ab
ba ∪ a
b
ϵ
ϵ
bb

Esempio 2

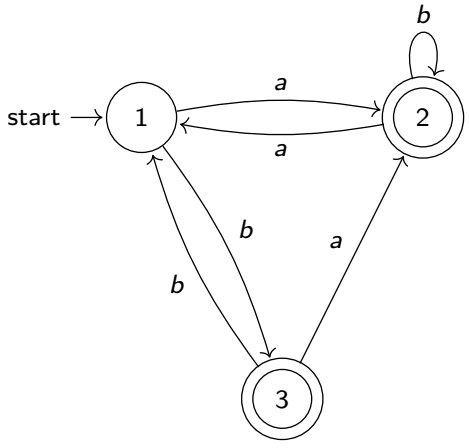

start
1
2
3
a
a
b
b
a
b

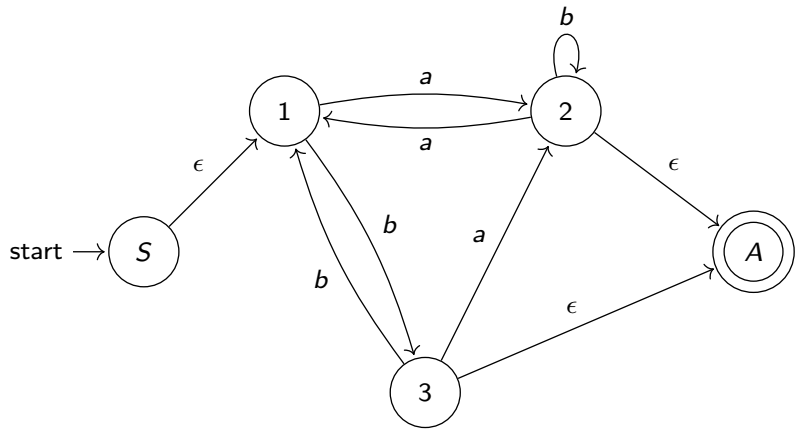

start
S
1
2
3
A
ϵ
a
a
b
b
a
ϵ
ϵ
b

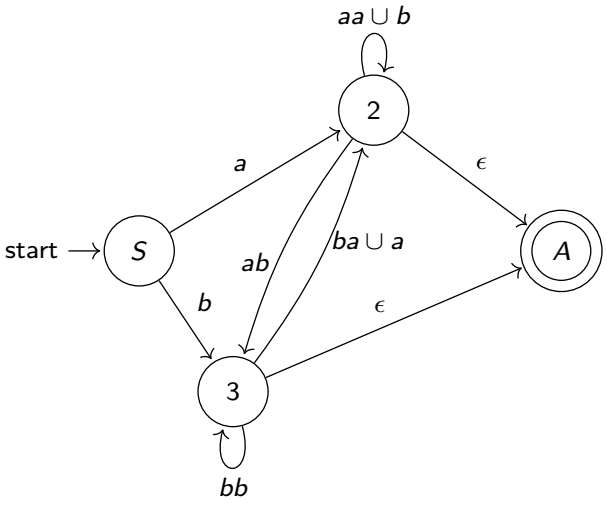

start
S
2
3
A
aa ∪ b
a
ab
ba ∪ a
b
ϵ
ϵ
bb

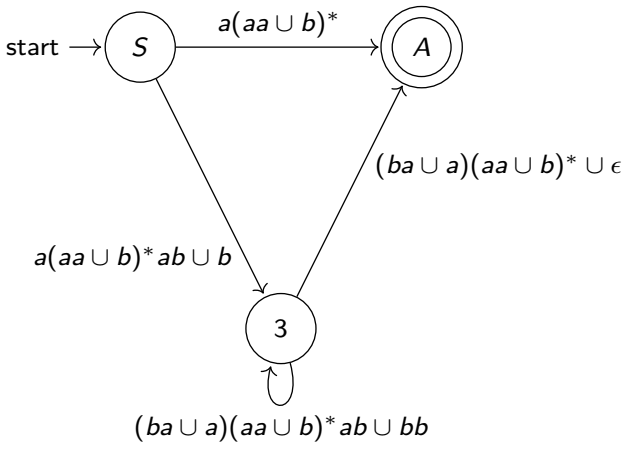

start
S
A
a(aa ∪ b)*
(ba ∪ a)(aa ∪ b)* ∪ ϵ
a(aa ∪ b)*ab ∪ b
3
(ba ∪ a)(aa ∪ b)*ab ∪ bb

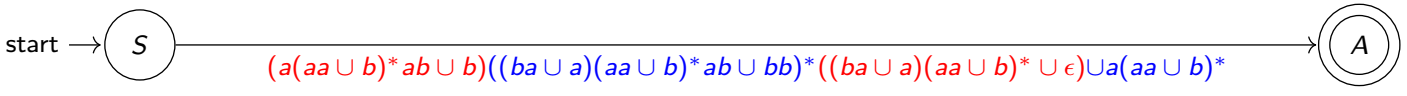
Esempio 2


start
1
2
3
a
a
b
b
a
b


start
S
1
2
3
A
ϵ
a
a
b
b
a
ϵ
ϵ


start
S
2
3
A
aa ∪ b
a
ϵ
ab
ba ∪ a
b
ϵ
bb


start
S
A
a(aa ∪ b)*
(ba ∪ a)(aa ∪ b)* ∪ ϵ
a(aa ∪ b)*ab ∪ b
3
(ba ∪ a)(aa ∪ b)*ab ∪ bb


start
S
A
(a(aa ∪ b)*ab ∪ b)((ba ∪ a)(aa ∪ b)*ab ∪ bb)*((ba ∪ a)(aa ∪ b)* ∪ ϵ)∪a(aa ∪ b)*

Tutti i linguaggi sono regolari?

La risposta è negativa. Consideriamo il seguente linguaggio:

$$L = \{a^n b^n | \ n \geq 0\}.$$

Una macchina che riconosca L, per essere in grado di controllare alla fine della lettura dell'input se il numero di a è uguale al numero di b, dovrebbe essere in grado di ricordare, mentre legge la stringa, quanti simboli uguali ad a ha letto.

Siccome n non è limitato, la macchina dovrebbe tener traccia di un numero illimitato di possibilità. Non può farlo perché ha un numero **finito** di stati.

Esiste una dimostrazione formale.

Pumping lemma

Teorema (Pumping lemma)

Se L è un linguaggio regolare, allora esiste una costante positiva p tale che per ogni stringa w in L di lunghezza $|w| \geq p$, esistono tre stringhe x, y, z con $w = xyz$ che soddisfano:

1. *per ogni $i \geq 0$, $xy^i z \in L$;*
2. $|y| > 0$;
3. $|xy| \leq p$.

Idea della dimostrazione.

Pumping lemma

Teorema (Pumping lemma)

Se L è un linguaggio regolare, allora esiste una costante positiva p tale che per ogni stringa w in L di lunghezza $|w| \geq p$, esistono tre stringhe x, y, z con $w = xyz$ che soddisfano:

1. *per ogni $i \geq 0$, $xy^i z \in L$;*
2. *$|y| > 0$;*
3. *$|xy| \leq p$.*

Idea della dimostrazione.

Sia M un automa che riconosce L e sia p il suo numero di stati.

Consideriamo una stringa w tale che $|w| \geq p$. Se w è accettata, esiste una sequenza $r_0, r_1, \ldots, r_{|w|}$ di $|w| + 1$ stati che parte dallo stato iniziale, legge la stringa un simbolo alla volta, e, seguendo la funzione di transizione, termina in uno stato accettante.

Il numero di stati della sequenza è maggiore del numero di stati di M: $|w| + 1 > p$. Quindi, esiste uno stato visitato due volte.

Pumping lemma

Sia r lo stato ripetuto. La stringa può essere divisa in tre parti: $w = xyz$, dove x porta dallo stato iniziale al primo r, y va dal primo al secondo r e z dal secondo r fino allo stato finale.

$$\overbrace{\underbrace{r_0, \ldots, r}_{x} \underbrace{\ldots, r}_{y} \underbrace{\ldots, r_{|w|}}_{z}}^{\text{sequenza di stati}}$$

Pumping lemma

Sia r lo stato ripetuto. La stringa può essere divisa in tre parti: $w = xyz$, dove x porta dallo stato iniziale al primo r, y va dal primo al secondo r e z dal secondo r fino allo stato finale.

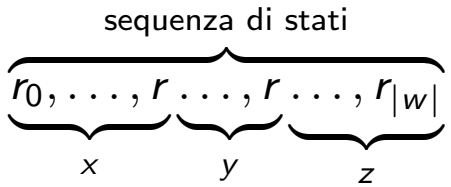
$$\overbrace{\underbrace{r_0, \ldots, r}_{x} \underbrace{\ldots, r}_{y} \underbrace{\ldots, r_{|w|}}_{z}}^{\text{sequenza di stati}}$$

Si ha: $|xy| \leq p$, altrimenti già nella sottostringa xy avremmo avuto uno stato ripetuto.

Inoltre $|y| > 0$ perché y è la parte della stringa che fa andare dalla prima occorrenza di r alla seconda (non può essere vuota).

Pumping lemma

Sia r lo stato ripetuto. La stringa può essere divisa in tre parti: $w = xyz$, dove x porta dallo stato iniziale al primo r, y va dal primo al secondo r e z dal secondo r fino allo stato finale.



Si ha: $|xy| \leq p$, altrimenti già nella sottostringa xy avremmo avuto uno stato ripetuto.

Inoltre $|y| > 0$ perché y è la parte della stringa che fa andare dalla prima occorrenza di r alla seconda (non può essere vuota).

Consideriamo l'input xy^iz per $i \geq 0$. L'automa M legge fino a x e arriva al primo r, poi leggendo y passa dal primo al secondo r. Poi prosegue la lettura con il secondo y e va ancora da r a r. E così via, fino a leggere z che porta dall'ultimo r fino allo stato finale. Quindi $xy^iz \in L$.

Pumping lemma

Sia r lo stato ripetuto. La stringa può essere divisa in tre parti: $w = xyz$, dove x porta dallo stato iniziale al primo r, y va dal primo al secondo r e z dal secondo r fino allo stato finale.

$$\overbrace{\underbrace{r_0, \ldots, r}_{x} \underbrace{\ldots, r}_{y} \underbrace{\ldots, r_{|w|}}_{z}}^{\text{sequenza di stati}}$$

Si ha: $|xy| \leq p$, altrimenti già nella sottostringa xy avremmo avuto uno stato ripetuto.

Inoltre $|y| > 0$ perché y è la parte della stringa che fa andare dalla prima occorrenza di r alla seconda (non può essere vuota).

Consideriamo l'input xy^iz per $i \geq 0$. L'automa M legge fino a x e arriva al primo r, poi leggendo y passa dal primo al secondo r. Poi prosegue la lettura con il secondo y e va ancora da r a r. E così via, fino a leggere z che porta dall'ultimo r fino allo stato finale. Quindi $xy^iz \in L$.

Tutte e tre le condizioni del teorema sono soddisfatte.

Applicazione del pumping lemma

Usiamo il pumping lemma per dimostrare il seguente teorema.

Teorema

Il linguaggio $C = \{w |\ w$ *ha lo stesso numero di simboli uguali a* 0 *e uguali a* $1\}$ *non è regolare.*

Dimostrazione.

Applicazione del pumping lemma

Usiamo il pumping lemma per dimostrare il seguente teorema.

Teorema

Il linguaggio $C = \{w |\ w$ *ha lo stesso numero di simboli uguali a* 0 *e uguali a* $1\}$ *non è regolare.*

Dimostrazione. Supponiamo per assurdo che C sia regolare. Allora vale il pumping lemma. Sia p la costante positiva p garantita dal pumping lemma.

Applicazione del pumping lemma

Usiamo il pumping lemma per dimostrare il seguente teorema.

Teorema

Il linguaggio $C = \{w |\ w$ ha lo stesso numero di simboli uguali a 0 e uguali a $1\}$ non è regolare.

Dimostrazione. Supponiamo per assurdo che C sia regolare. Allora vale il pumping lemma. Sia p la costante positiva p garantita dal pumping lemma.

Consideriamo la stringa $w = 0^p1^p$. Chiaramente, $w \in C$ e $|w| > p$. Il pumping lemma garantisce che w può essere divisa in tre parti $0^p1^p = xyz$, tali che

1. per ogni $i \geq 0$, $xy^iz \in C$;
2. $|y| > 0$;
3. $|xy| \leq p$.

La condizione 3 implica che xy è formata solo da zeri. Quindi anche y è formata solo da zeri (almeno uno per la condizione 2).

Applicazione del pumping lemma

Usiamo il pumping lemma per dimostrare il seguente teorema.

Teorema

Il linguaggio $C = \{w |$ w ha lo stesso numero di simboli uguali a 0 e uguali a $1\}$ non è regolare.

Dimostrazione. Supponiamo per assurdo che C sia regolare. Allora vale il pumping lemma. Sia p la costante positiva p garantita dal pumping lemma.

Consideriamo la stringa $w = 0^p1^p$. Chiaramente, $w \in C$ e $|w| > p$. Il pumping lemma garantisce che w può essere divisa in tre parti $0^p1^p = xyz$, tali che

1. per ogni $i \geq 0$, $xy^iz \in C$;
2. $|y| > 0$;
3. $|xy| \leq p$.

La condizione 3 implica che xy è formata solo da zeri. Quindi anche y è formata solo da zeri (almeno uno per la condizione 2).

Per $i > 1$, xy^iz ha un numero di zeri maggiore del numero di zeri di xyz, mentre il numero di 1 non è cambiato. Quindi, se $xyz \in C$, allora $xy^iz \notin C$. Ma la condizione 1 del pumping lemma afferma il contrario: **contraddizione**.

Ciò significa che C non può essere regolare.