STRUMENTI FORMALI PER LA BIOINFORMATICA

Automi finiti -
Parte 3. Automa minimale

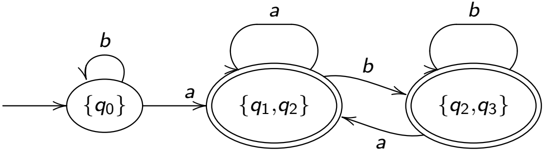
Ringraziamenti

Le figure sono prese dai libri (o dispense):

P. Degano, *Fondamenti di Informatica: Calcolabilità e Complessità*, Dispense, 2019.

J. Hopcroft, R. Motwani, J. Ullman , *Automi, Linguaggi e Calcolabilità*, Addison Wesley Pearson Education Italia s.r.l, Terza Edizione, 2009.

Michael Sipser, *Introduzione alla teoria della Computazione*, Apogeo Education, Maggioli Editore, 2016 (traduzione italiana di Introduction to the Theory of Computation, 3rd Edition).



Gli stati $\{q_1, q_2\}$ e $\{q_2, q_3\}$ possono essere "raggruppati".

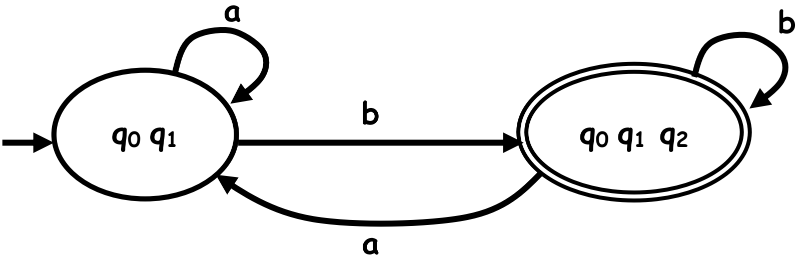


Figura: $\mathcal{A}$

Questo automa $\mathcal{A}$ ha il numero minimo di stati (tra tutti gli automi che riconoscono $L(\mathcal{A})$).

Per ogni linguaggio regolare L esiste uno speciale DFA che lo riconosce, chiamato l'automa minimale per L.

Per ogni linguaggio regolare L esiste uno speciale DFA che lo riconosce, chiamato l'automa minimale per L.

Il nome dipende dal fatto che se $L(\mathcal{A}) = L$, con $\mathcal{A}$ automa minimale, per ogni automa $\mathcal{B}$ tale che $L(\mathcal{B}) = L$,

Per ogni linguaggio regolare L esiste uno speciale DFA che lo riconosce, chiamato l'automa minimale per L.

Il nome dipende dal fatto che se $L(\mathcal{A}) = L$, con $\mathcal{A}$ automa minimale, per ogni automa $\mathcal{B}$ tale che $L(\mathcal{B}) = L$,

- $\mathcal{B}$ ha più stati di $\mathcal{A}$

Per ogni linguaggio regolare L esiste uno speciale DFA che lo riconosce, chiamato l'automa minimale per L.

Il nome dipende dal fatto che se $L(\mathcal{A}) = L$, con $\mathcal{A}$ automa minimale, per ogni automa $\mathcal{B}$ tale che $L(\mathcal{B}) = L$,

- $\mathcal{B}$ ha più stati di $\mathcal{A}$

  oppure

Per ogni linguaggio regolare L esiste uno speciale DFA che lo riconosce, chiamato l'automa minimale per L.

Il nome dipende dal fatto che se $L(\mathcal{A}) = L$, con $\mathcal{A}$ automa minimale, per ogni automa $\mathcal{B}$ tale che $L(\mathcal{B}) = L$,

- $\mathcal{B}$ ha più stati di $\mathcal{A}$

  oppure

- $\mathcal{B}$ ha lo stesso numero di stati di $\mathcal{A}$ e i due automi $\mathcal{A}$ e $\mathcal{B}$ sono uguali, a meno di una ridenominazione degli stati.

Per ogni linguaggio regolare L esiste uno speciale DFA che lo riconosce, chiamato l'automa minimale per L.

Il nome dipende dal fatto che se $L(\mathcal{A}) = L$, con $\mathcal{A}$ automa minimale, per ogni automa $\mathcal{B}$ tale che $L(\mathcal{B}) = L$,

- $\mathcal{B}$ ha più stati di $\mathcal{A}$

  oppure

- $\mathcal{B}$ ha lo stesso numero di stati di $\mathcal{A}$ e i due automi $\mathcal{A}$ e $\mathcal{B}$ sono uguali, a meno di una ridenominazione degli stati.

In altri termini, per ogni linguaggio regolare L esiste ed è unico l'automa minimale che riconosce L.

Automa minimale - Riassunto

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

Automa minimale - Riassunto

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da L.

- Esiste un algoritmo per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- Esiste un algoritmo per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- Questo algoritmo si basa sulla nozione di **stati equivalenti**.

Automa minimale - Riassunto

- Esiste un algoritmo per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- Questo algoritmo si basa sulla nozione di **stati equivalenti**.
- Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Diremo che $q, q' \in Q$ sono equivalenti se le stringhe che portano $\mathcal{A}$ da q a uno stato finale sono le stesse che portano $\mathcal{A}$ da q' a uno stato finale.

Automa minimale - Riassunto

- Esiste un algoritmo per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- Questo algoritmo si basa sulla nozione di **stati equivalenti**.
- Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Diremo che $q, q' \in Q$ sono equivalenti se le stringhe che portano $\mathcal{A}$ da q a uno stato finale sono le stesse che portano $\mathcal{A}$ da q' a uno stato finale.
- L'algoritmo divide Q in sottonsiemi formati da stati equivalenti, ognuno dei quali rappresenterà uno stato dell'automa minimale.

Automa minimale - Riassunto

- Esiste un algoritmo per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- Questo algoritmo si basa sulla nozione di **stati equivalenti**.
- Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Diremo che $q, q' \in Q$ sono equivalenti se le stringhe che portano $\mathcal{A}$ da q a uno stato finale sono le stesse che portano $\mathcal{A}$ da q' a uno stato finale.
- L'algoritmo divide Q in sottonsiemi formati da stati equivalenti, ognuno dei quali rappresenterà uno stato dell'automa minimale.
- Ogni stato nell'automa minimale sarà la fusione di tutti gli stati tra loro equivalenti.

Automa minimale - Riassunto

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da L.

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da L.
- A ogni DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ che riconosce L è possibile associare una relazione, questa volta su Σ^*, che "conta" il numero degli stati.

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da L.
- A ogni DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ che riconosce L è possibile associare una relazione, questa volta su Σ^*, che "conta" il numero degli stati.
- Diremo che due stringhe x, y sono equivalenti se leggendo l'una o l'altra a partire da q_0, $\mathcal{A}$ si troverà nello stesso stato.

Automa minimale - Riassunto

- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da L.
- A ogni DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ che riconosce L è possibile associare una relazione, questa volta su Σ^*, che "conta" il numero degli stati.
- Diremo che due stringhe x, y sono equivalenti se leggendo l'una o l'altra a partire da q_0, $\mathcal{A}$ si troverà nello stesso stato.
- Se ogni stato è raggiungibile da q_0, il numero delle classi è uguale al numero degli stati di $\mathcal{A}$: minimizzare gli stati equivale a minimizzare il numero delle classi.

Automa minimale - Riassunto

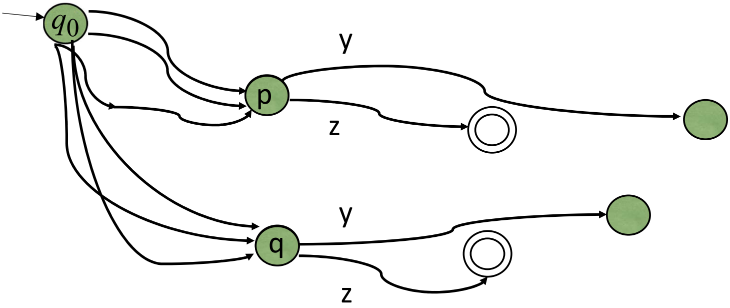
- È possibile costruire l'automa minimale di un linguaggio regolare L a partire da L.
- A ogni DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ che riconosce L è possibile associare una relazione, questa volta su Σ^*, che "conta" il numero degli stati.
- Diremo che due stringhe x, y sono equivalenti se leggendo l'una o l'altra a partire da q_0, $\mathcal{A}$ si troverà nello stesso stato.
- Se ogni stato è raggiungibile da q_0, il numero delle classi è uguale al numero degli stati di $\mathcal{A}$: minimizzare gli stati equivale a minimizzare il numero delle classi.
- Definiremo la relazione di equivalenza di Myhill-Nerode associata a L che ha questo numero minimo di classi. Il corrispondente automa, che ha come stati le classi, è l'automa minimale.

- Introdurremo tre relazioni di equivalenza.

- Introdurremo tre relazioni di equivalenza.
    1. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme degli stati di $\mathcal{A}$. L'automa minimale è ottenuto fondendo gli stati equivalenti.

DFA M

p e q stati equivalenti



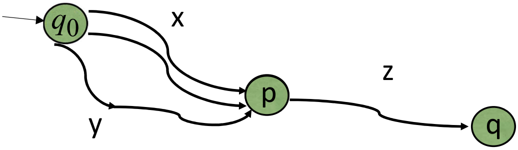
Automa minimale - Riassunto

- Introdurremo tre relazioni di equivalenza.

- Introdurremo tre relazioni di equivalenza.
  1. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme degli stati di $\mathcal{A}$. L'automa minimale è ottenuto fondendo gli stati equivalenti.

Automa minimale - Riassunto

- Introdurremo tre relazioni di equivalenza.
  1. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme degli stati di $\mathcal{A}$. L'automa minimale è ottenuto fondendo gli stati equivalenti.
  2. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme Σ^* delle stringhe sull'alfabeto dell'automa. Se ogni stato è raggiungibile da q_0, il numero delle classi è uguale al numero degli stati di $\mathcal{A}$.



x e y sono stringhe equivalenti rispetto ad R_A

- Introdurremo tre relazioni di equivalenza.

Automa minimale - Riassunto

- Introdurremo tre relazioni di equivalenza.
  1. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme degli stati di $\mathcal{A}$. L'automa minimale è ottenuto fondendo gli stati equivalenti.

Automa minimale - Riassunto

- Introdurremo tre relazioni di equivalenza.
  1. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme degli stati di $\mathcal{A}$. L'automa minimale è ottenuto fondendo gli stati equivalenti.
  2. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme Σ^* delle stringhe sull'alfabeto dell'automa. Se ogni stato è raggiungibile da q_0, il numero delle classi è uguale al numero degli stati di $\mathcal{A}$.

Automa minimale - Riassunto

- Introdurremo tre relazioni di equivalenza.
  1. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme degli stati di $\mathcal{A}$. L'automa minimale è ottenuto fondendo gli stati equivalenti.
  2. A partire da un DFA $\mathcal{A}$, una relazione di equivalenza nell'insieme Σ^* delle stringhe sull'alfabeto dell'automa. Se ogni stato è raggiungibile da q_0, il numero delle classi è uguale al numero degli stati di $\mathcal{A}$.
  3. Infine, definiremo la relazione di equivalenza di Myhill-Nerode associata a un linguaggio L. Se (e solo se) il linguaggio è regolare questa relazione ha un numero finito di classi di equivalenza. Costruiremo un automa, che ha come stati le classi, che risulterà essere l'automa minimale.

- Vedremo tre costruzioni dell'automa minimale.

- Vedremo tre costruzioni dell'automa minimale.
  1. L'algoritmo di costruzione dell'automa minimale basato sulla fusione degli stati equivalenti in un DFA $\mathcal{A}$.
     Input: un DFA $\mathcal{A}$
     Output: il DFA minimale equivalente ad $\mathcal{A}$.

Automa minimale - Riassunto

- Vedremo tre costruzioni dell'automa minimale.
  1. L'algoritmo di costruzione dell'automa minimale basato sulla fusione degli stati equivalenti in un DFA $\mathcal{A}$.
     Input: un DFA $\mathcal{A}$
     Output: il DFA minimale equivalente ad $\mathcal{A}$.
  2. La costruzione dell'automa minimale che ha come stati le classi di equivalenza rispetto alla relazione di equivalenza di Myhill-Nerode associata a un linguaggio regolare L.
     Input: un linguaggio regolare L
     Output: il DFA minimale che riconosce L.

- Vedremo tre costruzioni dell'automa minimale.
  1. L'algoritmo di costruzione dell'automa minimale basato sulla fusione degli stati equivalenti in un DFA $\mathcal{A}$.
     Input: un DFA $\mathcal{A}$
     Output: il DFA minimale equivalente ad $\mathcal{A}$.
  2. La costruzione dell'automa minimale che ha come stati le classi di equivalenza rispetto alla relazione di equivalenza di Myhill-Nerode associata a un linguaggio regolare L.
     Input: un linguaggio regolare L
     Output: il DFA minimale che riconosce L.
  3. L'algoritmo di Brzozowski di costruzione dell'automa minimale a partire da un DFA $\mathcal{A}$.
     Input: un DFA $\mathcal{A}$
     Output: il DFA minimale equivalente ad $\mathcal{A}$.

- Esistono algoritmi per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- Esistono algoritmi per costruire l'automa minimale di un linguaggio regolare L a partire da un DFA qualsiasi che riconosce L.

- L'algoritmo che descriveremo in generale non funziona se è applicato a un NFA che riconosce L.

1. Descrizione su un esempio dell'algoritmo.

1. Descrizione su un esempio dell’algoritmo.

2. Descrizione dell’algoritmo

1. Descrizione su un esempio dell'algoritmo.
2. Descrizione dell'algoritmo
3. Prova di correttezza dell'algoritmo e Teorema di Myhill-Nerode.

L'idea di fondo è quella di raggruppare in uno stato solo tutti quelli che si "comportano" nello stesso modo per poterli sostituire con un unico stato.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \hat{\delta}(q, w) \in F\}$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \stackrel{\wedge}{\delta}(q, w) \in F\}$.

Definizione di stati equivalenti. Siano $p, q \in Q$

$$p \equiv q \Leftrightarrow L_p = L_q$$

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \overset{\wedge}{\delta}(q, w) \in F\}$.

Definizione di stati equivalenti. Siano $p, q \in Q$

$$p \equiv q \Leftrightarrow L_p = L_q$$

In altri termini, $p \equiv q$ se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \overset{\wedge}{\delta}(q, w) \in F\}$.

Definizione di stati equivalenti. Siano $p, q \in Q$

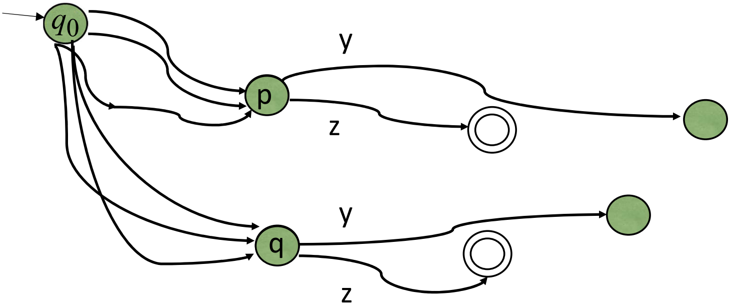
$$p \equiv q \Leftrightarrow L_p = L_q$$

In altri termini, $p \equiv q$ se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

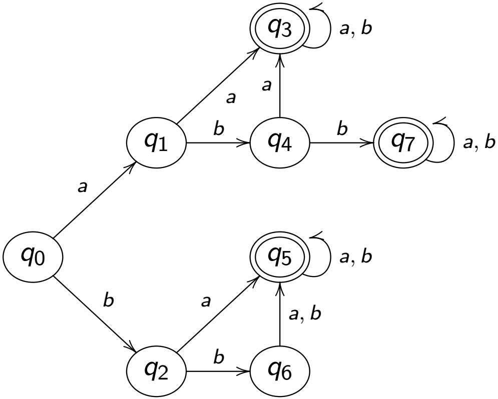
Due stati equivalenti sono anche chiamati indistinguibili.

DFA M

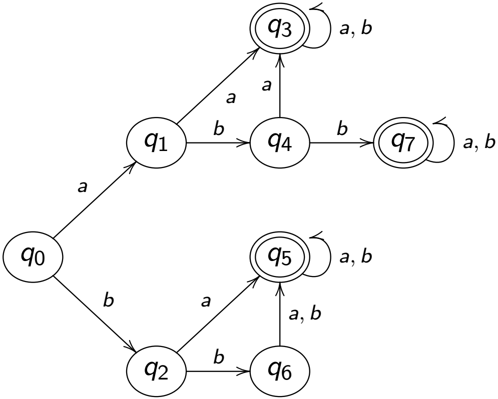
p e q stati equivalenti



Descrizione su un esempio dell'algoritmo

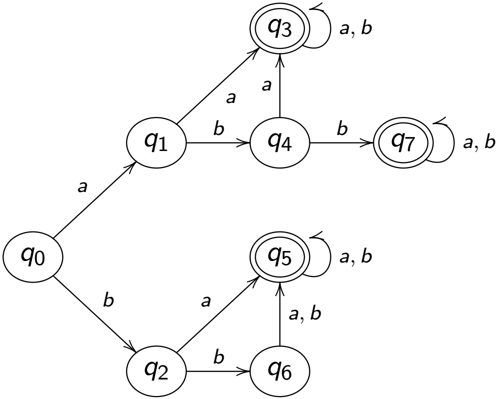


Descrizione su un esempio dell'algoritmo



$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

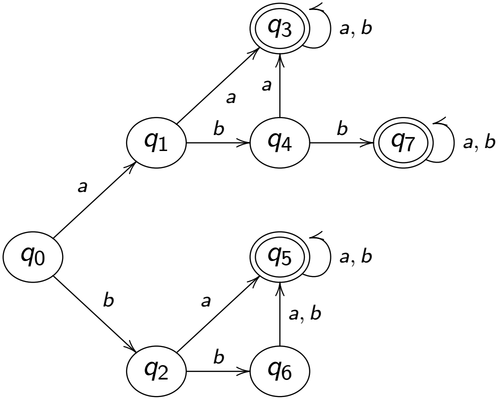
Descrizione su un esempio dell'algoritmo



$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

Descrizione su un esempio dell'algoritmo



$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

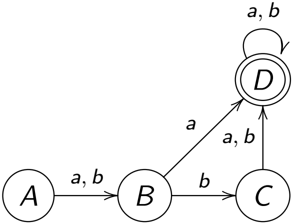
$$\Pi_3 = \{\{q_0\}, \{q_1, q_2\}, \{q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

Nuovo insieme di stati con A stato iniziale:

$$A = \{q_0\} \quad B = \{q_1, q_2\} \quad C = \{q_4, q_6\} \quad D = \{q_3, q_5, q_7\}$$

Nuovo insieme di stati con A stato iniziale:

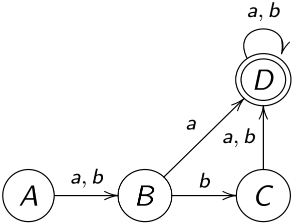
$$A = \{q_0\} \quad B = \{q_1, q_2\} \quad C = \{q_4, q_6\} \quad D = \{q_3, q_5, q_7\}$$



Descrizione su un esempio dell'algoritmo

Nuovo insieme di stati con A stato iniziale:

$$A = \{q_0\} \quad B = \{q_1, q_2\} \quad C = \{q_4, q_6\} \quad D = \{q_3, q_5, q_7\}$$



Definiamo $\overline{\delta}(X, \sigma) = Y$, dove X, Y sono nuovi stati e σ è un carattere, se $\delta(q, \sigma) = q'$ con $q \in X$ e $q' \in Y$ (per esempio $\overline{\delta}(B, b) = C$ perchè $\delta(q_1, b) = q_4$ e $q_1 \in B,\ q_4 \in C$).

Descriviamo formalmente l'algoritmo.

Descriviamo formalmente l'algoritmo.

Una partizione finita di un insieme X è una collezione $Y_1, \ldots, Y_n$ di sottoinsiemi di X tale che $Y_i \cap Y_j = \emptyset$, $1 \leq i \neq j \leq n$ e $\cup_{i=1}^{n} Y_i = X$.

Descriviamo formalmente l'algoritmo.

Una partizione finita di un insieme X è una collezione $Y_1, \ldots, Y_n$ di sottoinsiemi di X tale che $Y_i \cap Y_j = \emptyset$, $1 \leq i \neq j \leq n$ e $\cup_{i=1}^{n} Y_i = X$.

Una partizione di un insieme X è una collezione $(Y_i)_{i \in I}$ di sottoinsiemi di X tale che $Y_i \cap Y_j = \emptyset$, per $i, j \in I$, $i \neq j$ e $\cup_{i \in I} Y_i = X$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Due stati $p, q \in Q$ sono equivalenti se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Due stati $p, q \in Q$ sono equivalenti se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

Ora descriviamo un algoritmo ricorsivo. Quando questo algoritmo termina, otteniamo una partizione $\Pi = \{Q_1, \dots Q_n\}$ dell'insieme Q degli stati di $\mathcal{A}$ tale che:

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Due stati $p, q \in Q$ sono equivalenti se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

Ora descriviamo un algoritmo ricorsivo. Quando questo algoritmo termina, otteniamo una partizione $\Pi = \{Q_1, \ldots Q_n\}$ dell'insieme Q degli stati di $\mathcal{A}$ tale che:

- Tutti gli stati in Q_i sono equivalenti, $i = 1, \ldots, n$

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Due stati $p, q \in Q$ sono equivalenti se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

Ora descriviamo un algoritmo ricorsivo. Quando questo algoritmo termina, otteniamo una partizione $\Pi = \{Q_1, \dots Q_n\}$ dell'insieme Q degli stati di $\mathcal{A}$ tale che:

- Tutti gli stati in Q_i sono equivalenti, $i = 1, \dots, n$
- Per ogni i, j, $1 \leq i \neq j \leq n$, se $p \in Q_i$ e $q \in Q_j$ allora p e q non sono equivalenti.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Due stati $p, q \in Q$ sono equivalenti se, per ogni $w \in \Sigma^*$, $\hat{\delta}(p, w)$ è uno stato finale se e solo se $\hat{\delta}(q, w)$ è uno stato finale.

Ora descriviamo un algoritmo ricorsivo. Quando questo algoritmo termina, otteniamo una partizione $\Pi = \{Q_1, \dots Q_n\}$ dell'insieme Q degli stati di $\mathcal{A}$ tale che:

- Tutti gli stati in Q_i sono equivalenti, $i = 1, \dots, n$
- Per ogni i, j, $1 \leq i \neq j \leq n$, se $p \in Q_i$ e $q \in Q_j$ allora p e q non sono equivalenti.

L'algoritmo individua gli stati p, q NON equivalenti, chiamati per questo distinguibili.

L'algoritmo per individuare gli stati distinguibili in $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ è ricorsivo.

Algoritmo per individuare gli stati distinguibili

L'algoritmo per individuare gli stati distinguibili in $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ è ricorsivo.

Algoritmo $\mathcal{A}lg$

L'algoritmo per individuare gli stati distinguibili in $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ è ricorsivo.

Algoritmo $\mathcal{A}lg$

PASSO BASE: Se $p \in F$ e $q \notin F$, allora p e q sono stati distinguibili.

Algoritmo per individuare gli stati distinguibili

L'algoritmo per individuare gli stati distinguibili in $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ è ricorsivo.

Algoritmo $\mathcal{A}lg$

PASSO BASE: Se $p \in F$ e $q \notin F$, allora p e q sono stati distinguibili.

PASSO RICORSIVO: Siano p, q due stati tali che, per $a \in \Sigma$, $r = \delta(p, a)$ ed $s = \delta(q, a)$ siano distinguibili. Allora p e q sono stati distinguibili.

Iniziamo ripartendo Q in due sottoinsiemi usando il passo base.

Algoritmo per individuare gli stati distinguibili

Iniziamo ripartendo Q in due sottoinsiemi usando il passo base.

Ricorsivamente, a ciascuno dei sottoinsiemi ottenuti a un passo precedente applichiamo nuovamente (il passo ricorsivo di) $\mathcal{A}lg$. A ogni passo la regola del passo ricorsivo può essere applicata a un numero finito di coppie (l'insieme Q è finito) e un numero finito di volte a ogni coppia (al più $|\Sigma|$ volte).

Algoritmo per individuare gli stati distinguibili

Iniziamo ripartendo Q in due sottoinsiemi usando il passo base.

Ricorsivamente, a ciascuno dei sottoinsiemi ottenuti a un passo precedente applichiamo nuovamente (il passo ricorsivo di) $\mathcal{A}lg$. A ogni passo la regola del passo ricorsivo può essere applicata a un numero finito di coppie (l'insieme Q è finito) e un numero finito di volte a ogni coppia (al più $|\Sigma|$ volte).

Se tale applicazione non produce un cambiamento negli insiemi a cui lo abbiamo applicato, l'algoritmo termina.

Algoritmo per individuare gli stati distinguibili

Iniziamo ripartendo Q in due sottoinsiemi usando il passo base.

Ricorsivamente, a ciascuno dei sottoinsiemi ottenuti a un passo precedente applichiamo nuovamente (il passo ricorsivo di) $\mathcal{A}lg$. A ogni passo la regola del passo ricorsivo può essere applicata a un numero finito di coppie (l'insieme Q è finito) e un numero finito di volte a ogni coppia (al più $|\Sigma|$ volte).

Se tale applicazione non produce un cambiamento negli insiemi a cui lo abbiamo applicato, l'algoritmo termina.

L'algoritmo termina sempre. (Caso peggiore: tutti gli insiemi della partizione sono singoletti).

Teorema

L'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$. Quindi l'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se p e q non sono equivalenti.

Teorema

L'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$. Quindi l'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se p e q non sono equivalenti.

Teorema

L'algoritmo $\mathcal{A}lg$ non distingue p e q se e solo se p e q sono equivalenti.

Algoritmo per individuare gli stati distinguibili

Teorema

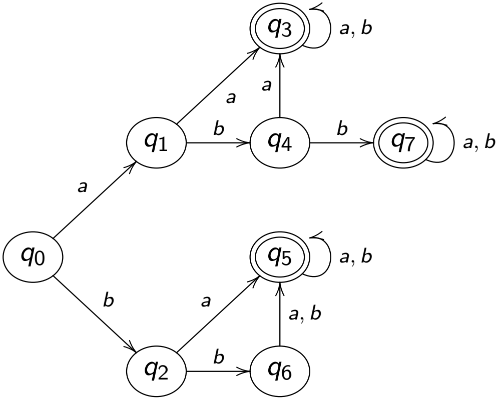
L'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se esiste $w \in \Sigma^*$ tale che $\hat{\delta}(q, w) \in F$ e $\hat{\delta}(p, w) \notin F$. Quindi l'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se p e q non sono equivalenti.

Teorema

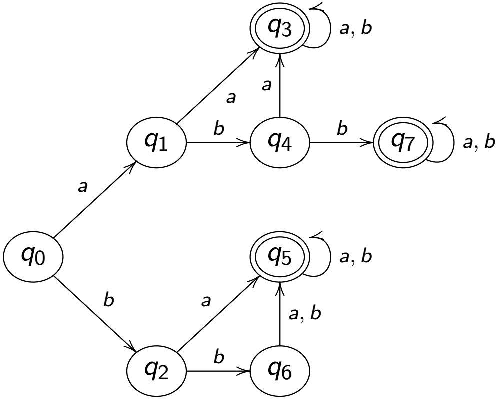
L'algoritmo $\mathcal{A}lg$ non distingue p e q se e solo se p e q sono equivalenti.

Quindi la partizione $\Pi = \{Q_1, \ldots Q_n\}$ di Q che otteniamo alla fine dell'algoritmo è tale che p e q sono equivalenti se e solo se esiste i, $1 \leq i \leq n$, tale che $p \in Q_i$ e $q \in Q_i$.

Algoritmo per individuare gli stati distinguibili

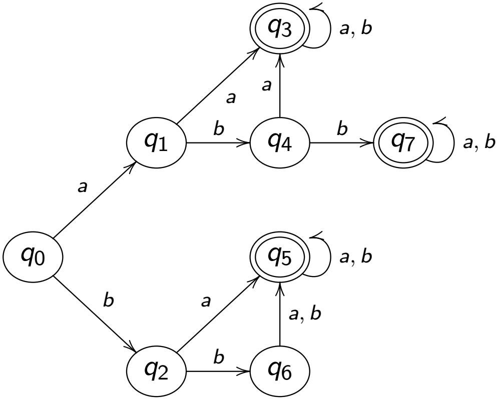


Algoritmo per individuare gli stati distinguibili



$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

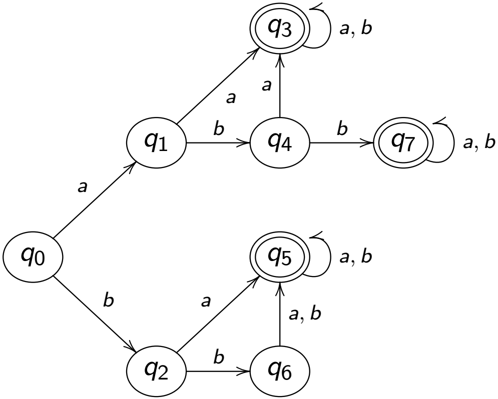
Algoritmo per individuare gli stati distinguibili



$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

Algoritmo per individuare gli stati distinguibili



$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

$$\Pi_3 = \{\{q_0\}, \{q_1, q_2\}, \{q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

Algoritmo per individuare gli stati distinguibili

- Applicando il passo base:

Algoritmo per individuare gli stati distinguibili

- Applicando il passo base:

$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

Algoritmo per individuare gli stati distinguibili

- Applicando il passo base:

$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

- Poiché $\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_3 = \delta(q_4, a)$, $\delta(q_2, a) = q_5 = \delta(q_6, a)$

Algoritmo per individuare gli stati distinguibili

- Applicando il passo base:

$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

- Poiché $\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_3 = \delta(q_4, a)$, $\delta(q_2, a) = q_5 = \delta(q_6, a)$

$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

Algoritmo per individuare gli stati distinguibili

- Applicando il passo base:

$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

- Poiché $\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_3 = \delta(q_4, a)$, $\delta(q_2, a) = q_5 = \delta(q_6, a)$

$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

- Poiché $\delta(q_1, b) = q_4$, $\delta(q_2, b) = q_6$, $\delta(q_4, b) = q_7$, $\delta(q_6, b) = q_5$

Algoritmo per individuare gli stati distinguibili

- Applicando il passo base:

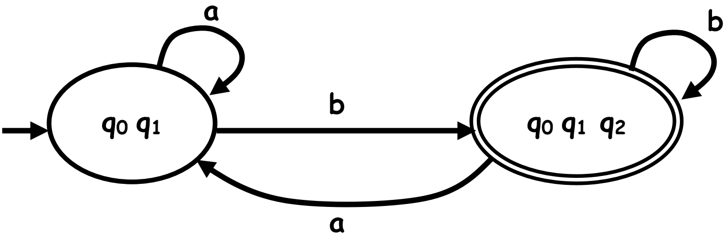
$$\Pi_1 = \{\{q_0, q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

- Poiché $\delta(q_0, a) = q_1$, $\delta(q_1, a) = q_3 = \delta(q_4, a)$, $\delta(q_2, a) = q_5 = \delta(q_6, a)$

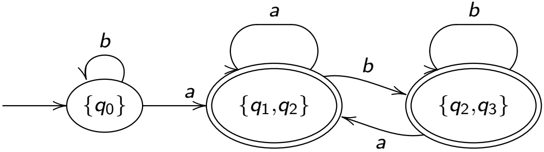
$$\Pi_2 = \{\{q_0\}, \{q_1, q_2, q_4, q_6\}, \{q_3, q_5, q_7\}\}$$

- Poiché $\delta(q_1, b) = q_4$, $\delta(q_2, b) = q_6$, $\delta(q_4, b) = q_7$, $\delta(q_6, b) = q_5$

$$\Pi_3 = \{\{q_0\}, \{q_1, q_2\}, \{q_4, q_6\}, \{q_3, q_5, q_7\}\}$$



L'algoritmo si ferma all'applicazione del passo base. Infatti il passo base divide l'insieme degli stati in due insiemi costituiti entrambi da un solo elemento. Quindi a nessuno dei due singoletti è possibile applicare il passo ricorsivo.



Poniamo $A = \{q_0\}$, $B = \{q_1, q_2\}$ e $C = \{q_2, q_3\}$.
Applicando il passo base otteniamo la partizione $\{\{A\}, \{B, C\}\}$.
Il passo ricorsivo non produce un cambiamento della partizione e quindi l'algoritmo termina.

Algoritmo per individuare gli stati equivalenti

Teorema

L'algoritmo $\mathcal{A}lg$ distingue p e q se e solo se esiste $w \in \Sigma^*$ tale che $\stackrel{\wedge}{\delta}(q, w) \in F$ e $\stackrel{\wedge}{\delta}(p, w) \notin F$.

Prova (cenni). Supponiamo che esista $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$. Proviamo per induzione su $|w|$ che l'algoritmo $\mathcal{A}lg$ distingue p e q.

Prova (cenni). Supponiamo che esista $w \in \Sigma^*$ tale che $\hat{\delta}(q, w) \in F$ e $\hat{\delta}(p, w) \notin F$. Proviamo per induzione su $|w|$ che l'algoritmo $\mathcal{A}lg$ distingue p e q.

Se $w = \epsilon$, allora $q \in F$, $p \notin F$ e il passo base distingue p e q.

Prova (cenni). Supponiamo che esista $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$. Proviamo per induzione su $|w|$ che l'algoritmo $\mathcal{A}lg$ distingue p e q.

Se $w = \epsilon$, allora $q \in F$, $p \notin F$ e il passo base distingue p e q.

Sia $w = a_1 \cdots a_n$, con $a_i \in \Sigma$, $i = 1, \ldots, n$, $n > 0$. Siano

$$q' = \delta(q, a_1), \quad p' = \delta(p, a_1), \quad w' = a_2 \cdots a_n.$$

Prova (cenni). Supponiamo che esista $w \in \Sigma^*$ tale che $\stackrel{\wedge}{\delta}(q, w) \in F$ e $\stackrel{\wedge}{\delta}(p, w) \notin F$. Proviamo per induzione su $|w|$ che l'algoritmo $\mathcal{A}lg$ distingue p e q.

Se $w = \epsilon$, allora $q \in F$, $p \notin F$ e il passo base distingue p e q.

Sia $w = a_1 \cdots a_n$, con $a_i \in \Sigma$, $i = 1, \ldots, n$, $n > 0$. Siano

$$q' = \delta(q, a_1), \quad p' = \delta(p, a_1), \quad w' = a_2 \cdots a_n.$$

È facile vedere che $\stackrel{\wedge}{\delta}(q', w') \in F$ e $\stackrel{\wedge}{\delta}(p', w') \notin F$, con $|w'| < n$.

Algoritmo per individuare gli stati equivalenti

Prova (cenni). Supponiamo che esista $w \in \Sigma^*$ tale che $\hat{\delta}(q, w) \in F$ e $\hat{\delta}(p, w) \notin F$. Proviamo per induzione su $|w|$ che l'algoritmo $\mathcal{A}lg$ distingue p e q.

Se $w = \epsilon$, allora $q \in F$, $p \notin F$ e il passo base distingue p e q.

Sia $w = a_1 \cdots a_n$, con $a_i \in \Sigma$, $i = 1, \ldots, n$, $n > 0$. Siano

$$q' = \delta(q, a_1), \quad p' = \delta(p, a_1), \quad w' = a_2 \cdots a_n.$$

È facile vedere che $\hat{\delta}(q', w') \in F$ e $\hat{\delta}(p', w') \notin F$, con $|w'| < n$.

Per ipotesi induttiva p', q' sono distinguibili e, in base al passo ricorsivo dell'algoritmo $\mathcal{A}lg$, anche p e q sono distinguibili.

Algoritmo per individuare gli stati equivalenti

Viceversa proviamo che se l'algoritmo $\mathcal{A}lg$ distingue p e q allora esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$, per induzione sul numero di iterazioni necessarie a distinguere p e q (induzione strutturale).

Viceversa proviamo che se l'algoritmo $\mathcal{A}lg$ distingue p e q allora esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$, per induzione sul numero di iterazioni necessarie a distinguere p e q (induzione strutturale).

Se l'algoritmo $\mathcal{A}lg$ distingue p e q nel passo base, allora solo uno dei due è in F. Quindi $\overset{\wedge}{\delta}(q, \epsilon) \in F$ e $\overset{\wedge}{\delta}(p, \epsilon) \notin F$ (o la relazione vale scambiando i ruoli di p e q) e p e q non sono equivalenti.

Viceversa proviamo che se l'algoritmo $\mathcal{A}lg$ distingue p e q allora esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$, per induzione sul numero di iterazioni necessarie a distinguere p e q (induzione strutturale).

Se l'algoritmo $\mathcal{A}lg$ distingue p e q nel passo base, allora solo uno dei due è in F. Quindi $\overset{\wedge}{\delta}(q, \epsilon) \in F$ e $\overset{\wedge}{\delta}(p, \epsilon) \notin F$ (o la relazione vale scambiando i ruoli di p e q) e p e q non sono equivalenti.

Supponiamo che l'algoritmo $\mathcal{A}lg$ distingue p e q dopo $k > 1$ iterazioni. Allora esistono $a \in \Sigma$, $r = \delta(p, a)$ ed $s = \delta(q, a)$ tali che r, s sono distinguibili.

Algoritmo per individuare gli stati equivalenti

Viceversa proviamo che se l'algoritmo $\mathcal{A}lg$ distingue p e q allora esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$, per induzione sul numero di iterazioni necessarie a distinguere p e q (induzione strutturale).

Se l'algoritmo $\mathcal{A}lg$ distingue p e q nel passo base, allora solo uno dei due è in F. Quindi $\overset{\wedge}{\delta}(q, \epsilon) \in F$ e $\overset{\wedge}{\delta}(p, \epsilon) \notin F$ (o la relazione vale scambiando i ruoli di p e q) e p e q non sono equivalenti.

Supponiamo che l'algoritmo $\mathcal{A}lg$ distingue p e q dopo $k > 1$ iterazioni. Allora esistono $a \in \Sigma$, $r = \delta(p, a)$ ed $s = \delta(q, a)$ tali che r, s sono distinguibili.

Per ipotesi induttiva, esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(r, w) \in F$ e $\overset{\wedge}{\delta}(s, w) \notin F$ (o la relazione vale scambiando i ruoli di r e s).

Viceversa proviamo che se l'algoritmo $\mathcal{A}lg$ distingue p e q allora esiste $w \in \Sigma^*$ tale che $\stackrel{\wedge}{\delta}(q, w) \in F$ e $\stackrel{\wedge}{\delta}(p, w) \notin F$, per induzione sul numero di iterazioni necessarie a distinguere p e q (induzione strutturale).

Se l'algoritmo $\mathcal{A}lg$ distingue p e q nel passo base, allora solo uno dei due è in F. Quindi $\stackrel{\wedge}{\delta}(q, \epsilon) \in F$ e $\stackrel{\wedge}{\delta}(p, \epsilon) \notin F$ (o la relazione vale scambiando i ruoli di p e q) e p e q non sono equivalenti.

Supponiamo che l'algoritmo $\mathcal{A}lg$ distingue p e q dopo $k > 1$ iterazioni. Allora esistono $a \in \Sigma$, $r = \delta(p, a)$ ed $s = \delta(q, a)$ tali che r, s sono distinguibili.

Per ipotesi induttiva, esiste $w \in \Sigma^*$ tale che $\stackrel{\wedge}{\delta}(r, w) \in F$ e $\stackrel{\wedge}{\delta}(s, w) \notin F$ (o la relazione vale scambiando i ruoli di r e s).

È facile vedere che $\stackrel{\wedge}{\delta}(p, aw) \in F$ e $\stackrel{\wedge}{\delta}(q, aw) \notin F$.

Viceversa proviamo che se l'algoritmo $\mathcal{A}lg$ distingue p e q allora esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(q, w) \in F$ e $\overset{\wedge}{\delta}(p, w) \notin F$, per induzione sul numero di iterazioni necessarie a distinguere p e q (induzione strutturale).

Se l'algoritmo $\mathcal{A}lg$ distingue p e q nel passo base, allora solo uno dei due è in F. Quindi $\overset{\wedge}{\delta}(q, \epsilon) \in F$ e $\overset{\wedge}{\delta}(p, \epsilon) \notin F$ (o la relazione vale scambiando i ruoli di p e q) e p e q non sono equivalenti.

Supponiamo che l'algoritmo $\mathcal{A}lg$ distingue p e q dopo $k > 1$ iterazioni. Allora esistono $a \in \Sigma$, $r = \delta(p, a)$ ed $s = \delta(q, a)$ tali che r, s sono distinguibili.

Per ipotesi induttiva, esiste $w \in \Sigma^*$ tale che $\overset{\wedge}{\delta}(r, w) \in F$ e $\overset{\wedge}{\delta}(s, w) \notin F$ (o la relazione vale scambiando i ruoli di r e s).

È facile vedere che $\overset{\wedge}{\delta}(p, aw) \in F$ e $\overset{\wedge}{\delta}(q, aw) \notin F$.

Questo conclude la prova.

Abbiamo dato un algoritmo per il calcolo degli stati equivalenti in un automa.

Abbiamo dato un algoritmo per il calcolo degli stati equivalenti in un automa.

Non abbiamo ancora definito l'automa minimale associato a un linguaggio regolare.

Relazione

Una relazione (binaria) R su un insieme S è un insieme di coppie di elementi di S.

Una relazione (binaria) R su un insieme S è un insieme di coppie di elementi di S.

Equivalentemente, $R \subseteq S \times S$.

Relazione

Una relazione (binaria) R su un insieme S è un insieme di coppie di elementi di S.

Equivalentemente, $R \subseteq S \times S$.

Scriveremo aRb se $(a, b) \in R$.

Una relazione (binaria) R su un insieme S è

Una relazione (binaria) R su un insieme S è

riflessiva se aRa, per ogni $a \in S$

Una relazione (binaria) R su un insieme S è

riflessiva se aRa, per ogni $a \in S$

simmetrica se aRb implica bRa

Relazione di equivalenza

Una relazione (binaria) R su un insieme S è

riflessiva se aRa, per ogni $a \in S$

simmetrica se aRb implica bRa

transitiva se aRb e bRc implica aRc

Relazione di equivalenza

Una relazione (binaria) R su un insieme S è

riflessiva se aRa, per ogni $a \in S$

simmetrica se aRb implica bRa

transitiva se aRb e bRc implica aRc

R è una relazione di equivalenza se R è riflessiva, simmetrica e transitiva.

- A ogni relazione di equivalenza R su S è associata una partizione in un numero (non necessariamente finito) di classi di equivalenza.

- A ogni relazione di equivalenza R su S è associata una partizione in un numero (non necessariamente finito) di classi di equivalenza.

  Le classi di equivalenza sono disgiunte e la loro unione fornisce come risultato S.

- A ogni relazione di equivalenza R su S è associata una partizione in un numero (non necessariamente finito) di classi di equivalenza.

  Le classi di equivalenza sono disgiunte e la loro unione fornisce come risultato S.

- Una classe di equivalenza E con rappresentante $x \in S$ è

$$E = \{y \in S \mid xRy\}.$$

- A ogni relazione di equivalenza R su S è associata una partizione in un numero (non necessariamente finito) di classi di equivalenza.

  Le classi di equivalenza sono disgiunte e la loro unione fornisce come risultato S.

- Una classe di equivalenza E con rappresentante $x \in S$ è

$$E = \{y \in S \mid xRy\}.$$

- La classe E viene anche denotata $[x]$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \hat{\delta}(q, w) \in F\}$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \overset{\wedge}{\delta}(q, w) \in F\}$.

Definizione di stati equivalenti. Siano $p, q \in Q$

$$p \equiv q \Leftrightarrow L_p = L_q$$

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \overset{\wedge}{\delta}(q, w) \in F\}$.

Definizione di stati equivalenti. Siano $p, q \in Q$

$$p \equiv q \Leftrightarrow L_p = L_q$$

In altri termini, $p \equiv q$ se, per ogni $w \in \Sigma^*$, $\overset{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\overset{\wedge}{\delta}(q, w)$ è uno stato finale.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA.

Per ogni $q \in Q$, sia $L_q = \{w \in \Sigma^* \mid \stackrel{\wedge}{\delta}(q, w) \in F\}$.

Definizione di stati equivalenti. Siano $p, q \in Q$

$$p \equiv q \Leftrightarrow L_p = L_q$$

In altri termini, $p \equiv q$ se, per ogni $w \in \Sigma^*$, $\stackrel{\wedge}{\delta}(p, w)$ è uno stato finale se e solo se $\stackrel{\wedge}{\delta}(q, w)$ è uno stato finale.

Due stati equivalenti sono anche chiamati indistinguibili.

Stati equivalenti

La relazione $\equiv$ è una relazione di equivalenza su Q.

La relazione $\equiv$ è una relazione di equivalenza su Q.

$[q]$ denoterà la classe di equivalenza di q rispetto a $\equiv$

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Sia $q \in Q$.

Stati accessibili

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Sia $q \in Q$.

Uno stato q è accessibile da q_0 se esiste una stringa $w \in \Sigma^*$ tale che $\stackrel{\wedge}{\delta}(q_0, w) = q$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Sia $q \in Q$.

Uno stato q è accessibile da q_0 se esiste una stringa $w \in \Sigma^*$ tale che $\hat{\delta}(q_0, w) = q$.

Esistono algoritmi per eliminare da $\mathcal{A}$ gli stati non accessibili da q_0. Se in $\mathcal{A}$ si eliminano gli stati non accessibili da q_0 si ottiene un automa equivalente ad $\mathcal{A}$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

Costruzione dell'automa minimale per L

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

L'automa minimale per L è

$$\mathcal{B} = (Q_m, \Sigma, \delta_m, [q_0], F_m)$$

Costruzione dell'automa minimale per L

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.
L'automa minimale per L è

$$\mathcal{B} = (Q_m, \Sigma, \delta_m, [q_0], F_m)$$

dove

$$\begin{array}{rcl} Q_m & = & \{[q] \mid q \in Q \text{ e } q \text{ è accessibile da } q_0\} \\ F_m & = & \{[q] \mid q \in F\} \\ \delta_m([q], a) & = & [\delta(q, a)], \quad \text{per ogni } [q] \in Q_m, a \in \Sigma \end{array}$$

- Nota: la definizione di δ_m è ben posta.

- Nota: la definizione di δ_m è ben posta.
  Infatti, per ogni $q, q' \in Q$, $a \in \Sigma$,

$$q \equiv q' \Rightarrow \delta(q,a) \equiv \delta(q',a) \Rightarrow [\delta(q,a)] = [\delta(q',a)]$$

- Nota: la definizione di δ_m è ben posta.
  Infatti, per ogni $q, q' \in Q$, $a \in \Sigma$,

$$q \equiv q' \Rightarrow \delta(q,a) \equiv \delta(q',a) \Rightarrow [\delta(q,a)] = [\delta(q',a)]$$

- Nota: per induzione su $|w|$ si può dimostrare che risulta

$$\overset{\wedge}{\delta}_m([q],w) = [\overset{\wedge}{\delta}(q,w)]$$

per ogni $q \in Q$ e $w \in \Sigma^*$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

$$L(\mathcal{B}) = L(\mathcal{A})$$

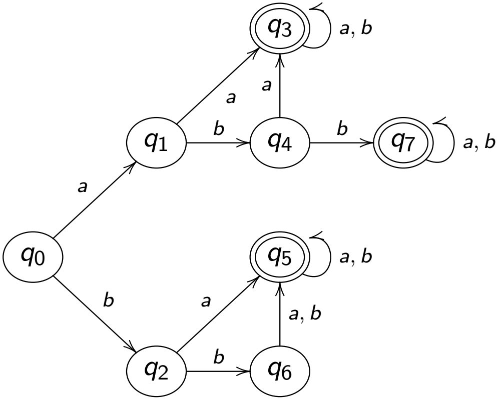
Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

$$L(\mathcal{B}) = L(\mathcal{A})$$

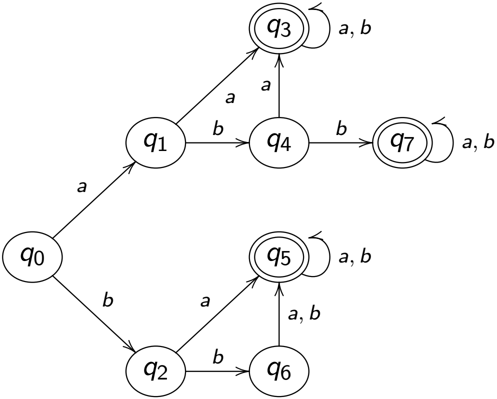
Infatti:

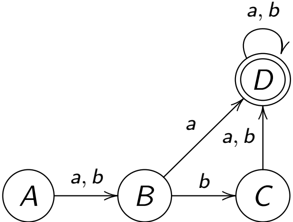
$$\begin{aligned} w \in L(\mathcal{B}) \quad &\Leftrightarrow \quad \hat{\delta}_m([q_0], w) = [\hat{\delta}(q_0, w)] \in F_m \\ &\Leftrightarrow \quad \hat{\delta}(q_0, w) \in F \Leftrightarrow w \in L(\mathcal{A}) \end{aligned}$$

$\mathcal{A}$ (stato iniziale q_0) e $\mathcal{B}$ (stato iniziale A) minimale



$\mathcal{A}$ (stato iniziale q_0) e $\mathcal{B}$ (stato iniziale A) minimale





Perché $\mathcal{B}$ è l'automa minimale di $L(\mathcal{A})$?

Prova di correttezza dell'algoritmo e Teorema di Myhill-Nerode.

Sia R una relazione di equivalenza su S.

Sia R una relazione di equivalenza su S.

L'indice di R è il numero delle classi di equivalenza di R. Può essere infinito.

Sia R una relazione di equivalenza su S.

L'indice di R è il numero delle classi di equivalenza di R. Può essere infinito.

Se l'indice di R è finito diciamo che R è di indice finito.

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- **Esempio.** Sia $S = \{a, b, c, d, e, f, g\}$. Siano R_1, R_2 due relazioni di equivalenza su S e siano π_1 e π_2 le corrispondenti partizioni associate

$$\pi_1 = \{\{a, b\}, \{c\}, \{d, e\}, \{f, g\}\},$$

$$\pi_2 = \{\{a, b, c\}, \{d, e\}, \{f, g\}\}$$

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- **Esempio.** Sia $S = \{a, b, c, d, e, f, g\}$. Siano R_1, R_2 due relazioni di equivalenza su S e siano π_1 e π_2 le corrispondenti partizioni associate

$$\pi_1 = \{\{a, b\}, \{c\}, \{d, e\}, \{f, g\}\},$$

$$\pi_2 = \{\{a, b, c\}, \{d, e\}, \{f, g\}\}$$

  R_1 è un raffinamento di R_2.

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

  Quindi xE_ny significa $x \equiv y \pmod{n}$
  (cioè $x - y$ è un multiplo di n).

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

  Quindi xE_ny significa $x \equiv y \pmod{n}$
  (cioè $x - y$ è un multiplo di n).

  E_n ha n classi, corrispondenti alle classi dei resti $\pmod{n}$.

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

  Quindi xE_ny significa $x \equiv y \pmod{n}$ (cioè $x - y$ è un multiplo di n).

  E_n ha n classi, corrispondenti alle classi dei resti $\pmod{n}$.

  Siccome $x \equiv y \pmod{6}$ implica $x \equiv y \pmod{3}$, la relazione E_6 è un raffinamento di E_3.

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

  Quindi xE_ny significa $x \equiv y \pmod{n}$
  (cioè $x - y$ è un multiplo di n).

  E_n ha n classi, corrispondenti alle classi dei resti $\pmod{n}$.

  Siccome $x \equiv y \pmod 6$ implica $x \equiv y \pmod 3$, la relazione E_6 è un raffinamento di E_3.

  E_3 ha tre classi: $[0]_3, [1]_3, [2]_2$.

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

  Quindi xE_ny significa $x \equiv y \pmod{n}$
  (cioè $x - y$ è un multiplo di n).

  E_n ha n classi, corrispondenti alle classi dei resti $\pmod{n}$.

  Siccome $x \equiv y \pmod{6}$ implica $x \equiv y \pmod{3}$, la relazione E_6 è un raffinamento di E_3.

  E_3 ha tre classi: $[0]_3, [1]_3, [2]_2$.

  E_6 ha sei classi: $[0]_6, [1]_6, [2]_6, [3]_6, [4]_6, [5]_6$.

- **Esempio.** Sia E_n la congruenza modulo n nell'insieme $\mathbb{Z}$ degli interi.

  Quindi xE_ny significa $x \equiv y \pmod{n}$
  (cioè $x - y$ è un multiplo di n).

  E_n ha n classi, corrispondenti alle classi dei resti $\pmod{n}$.

  Siccome $x \equiv y \pmod{6}$ implica $x \equiv y \pmod{3}$, la relazione E_6 è un raffinamento di E_3.

  E_3 ha tre classi: $[0]_3, [1]_3, [2]_2$.

  E_6 ha sei classi: $[0]_6, [1]_6, [2]_6, [3]_6, [4]_6, [5]_6$.

  Ogni classe di equivalenza di E_3 è unione di due delle classi di equivalenza di E_6.

Esempio. E_3 ha tre classi:

$$[0]_3 = \{3k \mid k \in \mathbb{N}, k \geq 0\},$$

$$[1]_3 = \{3k+1 \mid k \in \mathbb{N}, k \geq 0\},$$

$$[2]_3 = \{3k+2 \mid k \in \mathbb{N}, k \geq 0\}$$

Esempio. E_3 ha tre classi:

$$[0]_3 = \{3k \mid k \in \mathbb{N}, k \geq 0\},$$

$$[1]_3 = \{3k + 1 \mid k \in \mathbb{N}, k \geq 0\},$$

$$[2]_3 = \{3k + 2 \mid k \in \mathbb{N}, k \geq 0\}$$

E_6 ha sei classi:

$$[0]_6 = \{6k \mid k \in \mathbb{N}, k \geq 0\} \subseteq [0]_3,$$

$$[1]_6 = \{6k + 1 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [1]_3,$$

$$[2]_6 = \{6k + 2 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [2]_3,$$

$$[3]_6 = \{6k + 3 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [0]_3,$$

$$[4]_6 = \{6k + 4 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [1]_3,$$

$$[5]_6 = \{6k + 5 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [2]_3$$

Esempio. E_3 ha tre classi:

$$[0]_3 = \{3k \mid k \in \mathbb{N}, k \geq 0\},$$

$$[1]_3 = \{3k+1 \mid k \in \mathbb{N}, k \geq 0\},$$

$$[2]_3 = \{3k+2 \mid k \in \mathbb{N}, k \geq 0\}$$

E_6 ha sei classi:

$$[0]_6 = \{6k \mid k \in \mathbb{N}, k \geq 0\} \subseteq [0]_3,$$

$$[1]_6 = \{6k+1 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [1]_3,$$

$$[2]_6 = \{6k+2 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [2]_3,$$

$$[3]_6 = \{6k+3 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [0]_3,$$

$$[4]_6 = \{6k+4 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [1]_3,$$

$$[5]_6 = \{6k+5 \mid k \in \mathbb{N}, k \geq 0\} \subseteq [2]_3$$

E_6 è un raffinamento di E_3.

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- Se R_1 è un raffinamento di R_2 allora ogni classe di equivalenza di R_2 è unione di alcune classi di equivalenza di R_1. Quindi, se R_1 ha indice finito ed $R_1 \neq R_2$, allora R_2 ha un minor numero di classi.

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- Se R_1 è un raffinamento di R_2 allora ogni classe di equivalenza di R_2 è unione di alcune classi di equivalenza di R_1. Quindi, se R_1 ha indice finito ed $R_1 \neq R_2$, allora R_2 ha un minor numero di classi.

  Infatti, denotiamo con $[x]_2$ la classe di x rispetto ad R_2 e con $[x]_1$ la classe di x rispetto ad R_1.

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- Se R_1 è un raffinamento di R_2 allora ogni classe di equivalenza di R_2 è unione di alcune classi di equivalenza di R_1. Quindi, se R_1 ha indice finito ed $R_1 \neq R_2$, allora R_2 ha un minor numero di classi.

  Infatti, denotiamo con $[x]_2$ la classe di x rispetto ad R_2 e con $[x]_1$ la classe di x rispetto ad R_1.

  Per ogni $y \in [x]_2$, risulta $[y]_1 \subseteq [x]_2$. Infatti, se $z \in [y]_1$ allora zR_1y implica zR_2y. Quindi $z \in [y]_2 = [x]_2$.

- Siano R_1, R_2 relazioni di equivalenza su S.
  R_1 è un raffinamento di R_2 se $R_1 \subseteq R_2$, cioè se

$$\forall x, y \in S \quad xR_1y \Rightarrow xR_2y$$

- Se R_1 è un raffinamento di R_2 allora ogni classe di equivalenza di R_2 è unione di alcune classi di equivalenza di R_1. Quindi, se R_1 ha indice finito ed $R_1 \neq R_2$, allora R_2 ha un minor numero di classi.

  Infatti, denotiamo con $[x]_2$ la classe di x rispetto ad R_2 e con $[x]_1$ la classe di x rispetto ad R_1.

  Per ogni $y \in [x]_2$, risulta $[y]_1 \subseteq [x]_2$. Infatti, se $z \in [y]_1$ allora zR_1y implica zR_2y. Quindi $z \in [y]_2 = [x]_2$.

  Allora $[x]_2 = \cup_{y \in [x]_2} [y]_1$.

Due relazioni di equivalenza in Σ^*

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.

La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$$

Due relazioni di equivalenza in Σ^*

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$$

$R_{\mathcal{A}}$ è una relazione di equivalenza e l'indice di $R_{\mathcal{A}}$ è al più $|Q|$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.

La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$R_{\mathcal{A}}$ è una relazione di equivalenza e l'indice di $R_{\mathcal{A}}$ è al più $|Q|$.

Infatti alla classe di x è associato lo stato $\overset{\wedge}{\delta}(q_0, x)$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_\mathcal{A}$ è definita da

$$xR_\mathcal{A}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$R_\mathcal{A}$ è una relazione di equivalenza e l'indice di $R_\mathcal{A}$ è al più $|Q|$.

Infatti alla classe di x è associato lo stato $\overset{\wedge}{\delta}(q_0, x)$.

Nota. Perché l'indice di $R_\mathcal{A}$ è al più $|Q|$ e non è sempre uguale a $|Q|$?

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_{\mathcal{A}}$ è definita da

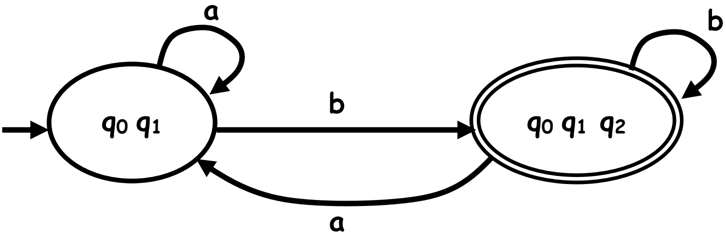
$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$R_{\mathcal{A}}$ è una relazione di equivalenza e l'indice di $R_{\mathcal{A}}$ è al più $|Q|$.

Infatti alla classe di x è associato lo stato $\overset{\wedge}{\delta}(q_0, x)$.

Nota. Perché l'indice di $R_{\mathcal{A}}$ è al più $|Q|$ e non è sempre uguale a $|Q|$?
Se q è uno stato di $\mathcal{A}$ non raggiungibile da q_0, cioè per il quale non esiste alcuna stringa z tale che $\overset{\wedge}{\delta}(q_0, z) = q$, a q non è associata alcuna classe.

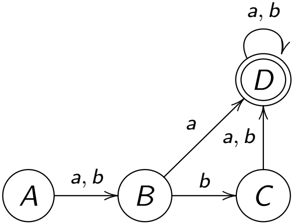
Nota. L'indice di $R_{\mathcal{A}}$ è uguale a $|Q|$ se e solo se ogni stato di $\mathcal{A}$ è raggiungibile da q_0.

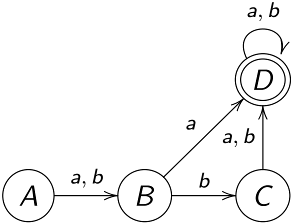


La relazione $R_{\mathcal{A}}$ per l'automa $\mathcal{A}$ in figura ha due classi:

- $[a]$ che contiene tutte le stringhe in $\{a, b\}^*a \cup \{\epsilon\}$
- $[b]$ che contiene tutte le stringhe in $\{a, b\}^*b$.

Le due classi sono individuate dai due stati.

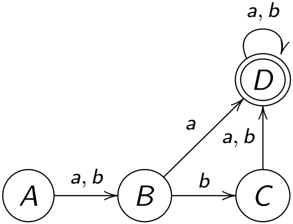

a, b
D
a
a, b
A
a, b
B
b
C



La relazione $R_{\mathcal{B}}$ per l'automa $\mathcal{B}$ in figura, con stato iniziale A, ha 4 classi:

- $[\epsilon]$ che contiene solo la parola vuota.
- $[a]$ che contiene le stringhe in $\{a, b\}$
- $[ab]$ che contiene le stringhe in $\{ab, bb\}$.
- $[aa]$ che contiene tutte le stringhe in
$\{aa, ba\}\{a, b\}^* \cup \{ab, bb\}\{a, b\}^+$

La relazione $R_{\mathcal{A}}$



La relazione $R_{\mathcal{B}}$ per l'automa $\mathcal{B}$ in figura, con stato iniziale A, ha 4 classi:

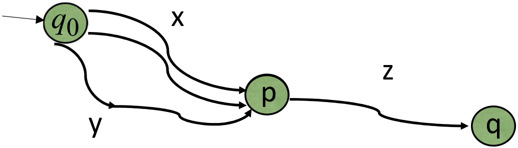
- $[\epsilon]$ che contiene solo la parola vuota.
- $[a]$ che contiene le stringhe in $\{a, b\}$
- $[ab]$ che contiene le stringhe in $\{ab, bb\}$.
- $[aa]$ che contiene tutte le stringhe in $\{aa, ba\}\{a, b\}^* \cup \{ab, bb\}\{a, b\}^+$

Le 4 classi sono associate ai 4 stati dell'automa.

Relazioni di equivalenza invarianti a destra

Siano $x, y \in \Sigma^*$. Una relazione di equivalenza R su Σ^* è invariante a destra (rispetto alla concatenazione) se xRy implica $xzRyz$ per ogni $z \in \Sigma^*$.

Relazioni di equivalenza invarianti a destra



x e y sono stringhe equivalenti rispetto ad R_A

Siano $x, y \in \Sigma^*$. Una relazione di equivalenza R su Σ^* è invariante a destra (rispetto alla concatenazione) se xRy implica $xzRyz$ per ogni $z \in \Sigma^*$.

Siano $x, y \in \Sigma^*$. Una relazione di equivalenza R su Σ^* è invariante a destra (rispetto alla concatenazione) se xRy implica $xzRyz$ per ogni $z \in \Sigma^*$.

$R_{\mathcal{A}}$ è invariante a destra. Supponiamo $xR_{\mathcal{A}}y$ e proviamo che $xzR_{\mathcal{A}}yz$.

Relazioni di equivalenza invarianti a destra

Siano $x, y \in \Sigma^*$. Una relazione di equivalenza R su Σ^* è invariante a destra (rispetto alla concatenazione) se xRy implica $xzRyz$ per ogni $z \in \Sigma^*$.

$R_{\mathcal{A}}$ è invariante a destra. Supponiamo $xR_{\mathcal{A}}y$ e proviamo che $xzR_{\mathcal{A}}yz$.

$$\begin{aligned}
\hat{\delta}(q_0, xz) &= \hat{\delta}(\hat{\delta}(q_0, x), z) \\
&= \hat{\delta}(\hat{\delta}(q_0, y), z) \\
&= \hat{\delta}(q_0, yz)
\end{aligned}$$

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.

La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

Due relazioni di equivalenza in Σ^*

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Sia $x \in L$, proviamo che la classe di x rispetto ad $R_{\mathcal{A}}$ è contenuta in L.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.
Sia $x \in L$, proviamo che la classe di x rispetto ad $R_{\mathcal{A}}$ è contenuta in L.
Se $x \in L$ allora $\overset{\wedge}{\delta}(q_0, x) \in F$. Se y è nella classe di x rispetto ad $R_{\mathcal{A}}$, allora $yR_{\mathcal{A}}x$. Quindi $\overset{\wedge}{\delta}(q_0, y) = \overset{\wedge}{\delta}(q_0, x) \in F$ e $y \in L$.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA. Siano $x, y \in \Sigma^*$.
La relazione $R_{\mathcal{A}}$ è definita da

$$xR_{\mathcal{A}}y \quad \Leftrightarrow \quad \overset{\wedge}{\delta}(q_0, x) = \overset{\wedge}{\delta}(q_0, y)$$

$L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Sia $x \in L$, proviamo che la classe di x rispetto ad $R_{\mathcal{A}}$ è contenuta in L.

Se $x \in L$ allora $\overset{\wedge}{\delta}(q_0, x) \in F$. Se y è nella classe di x rispetto ad $R_{\mathcal{A}}$, allora $yR_{\mathcal{A}}x$. Quindi $\overset{\wedge}{\delta}(q_0, y) = \overset{\wedge}{\delta}(q_0, x) \in F$ e $y \in L$.

In altri termini, siccome $x \in L \Leftrightarrow \overset{\wedge}{\delta}(q_0, x) \in F$ allora L è unione delle classi che corrispondono agli stati finali di $\mathcal{A}$.

Quindi, dato un DFA $\mathcal{A}$, la relazione $R_{\mathcal{A}}$ è invariante a destra e $L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Quindi, dato un DFA $\mathcal{A}$, la relazione $R_{\mathcal{A}}$ è invariante a destra e $L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Definiamo ora la relazione di equivalenza di Myhill-Nerode R_L associata a un linguaggio L (regolare o non regolare).

Quindi, dato un DFA $\mathcal{A}$, la relazione $R_{\mathcal{A}}$ è invariante a destra e $L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Definiamo ora la relazione di equivalenza di Myhill-Nerode R_L associata a un linguaggio L (regolare o non regolare).

Proveremo che:

Quindi, dato un DFA $\mathcal{A}$, la relazione $R_{\mathcal{A}}$ è invariante a destra e $L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Definiamo ora la relazione di equivalenza di Myhill-Nerode R_L associata a un linguaggio L (regolare o non regolare).

Proveremo che:

- R_L è invariante a destra e L è unione di classi di R_L.

Quindi, dato un DFA $\mathcal{A}$, la relazione $R_{\mathcal{A}}$ è invariante a destra e $L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Definiamo ora la relazione di equivalenza di Myhill-Nerode R_L associata a un linguaggio L (regolare o non regolare).

Proveremo che:

- R_L è invariante a destra e L è unione di classi di R_L.
- Ogni relazione di equivalenza E invariante a destra e tale che L sia unione di classi di E è un raffinamento di R_L.

Quindi, dato un DFA $\mathcal{A}$, la relazione $R_{\mathcal{A}}$ è invariante a destra e $L = L(\mathcal{A})$ è unione di classi della relazione $R_{\mathcal{A}}$.

Definiamo ora la relazione di equivalenza di Myhill-Nerode R_L associata a un linguaggio L (regolare o non regolare).

Proveremo che:

- R_L è invariante a destra e L è unione di classi di R_L.
- Ogni relazione di equivalenza E invariante a destra e tale che L sia unione di classi di E è un raffinamento di R_L.
- Se L è regolare, R_L ha indice finito e ha il minimo numero di classi tra tutte le relazioni di equivalenza E invarianti a destra e tali che L sia unione di classi di E.

Sia $L \subseteq \Sigma^*$, siano $x, y \in \Sigma^*$. Sia R_L definita da: xR_Ly se e solo se

$$\forall z \in \Sigma^* \ xz \in L \Leftrightarrow yz \in L$$

Due relazioni di equivalenza in Σ^*

Sia $L \subseteq \Sigma^*$, siano $x, y \in \Sigma^*$. Sia R_L definita da: xR_Ly se e solo se

$$\forall z \in \Sigma^* \ xz \in L \Leftrightarrow yz \in L$$

La relazione R_L è anche chiamata relazione di equivalenza di Myhill-Nerode.

Due relazioni di equivalenza in Σ^*

Sia $L \subseteq \Sigma^*$, siano $x, y \in \Sigma^*$. Sia R_L definita da: xR_Ly se e solo se

$$\forall z \in \Sigma^* \; xz \in L \Leftrightarrow yz \in L$$

La relazione R_L è anche chiamata relazione di equivalenza di Myhill-Nerode.

R_L è una relazione di equivalenza invariante a destra.
Supponiamo xR_Ly e proviamo che xuR_Lyu per ogni $u \in \Sigma^*$.

$$(xu)z \in L \;\Leftrightarrow\; x(uz) \in L \;\Leftrightarrow\; y(uz) \in L \;\Leftrightarrow\; (yu)z \in L$$

Sia $L \subseteq \Sigma^*$, siano $x, y \in \Sigma^*$. Sia R_L definita da: xR_Ly se e solo se

$$\forall z \in \Sigma^* \; xz \in L \Leftrightarrow yz \in L$$

Due relazioni di equivalenza in Σ^*

Sia $L \subseteq \Sigma^*$, siano $x, y \in \Sigma^*$. Sia R_L definita da: xR_Ly se e solo se

$$\forall z \in \Sigma^* \; xz \in L \Leftrightarrow yz \in L$$

L è unione di classi della relazione di Myhill-Nerode R_L, le classi che corrispondono agli elementi di L.

Due relazioni di equivalenza in Σ^*

Sia $L \subseteq \Sigma^*$, siano $x, y \in \Sigma^*$. Sia R_L definita da: xR_Ly se e solo se

$$\forall z \in \Sigma^* \; xz \in L \Leftrightarrow yz \in L$$

L è unione di classi della relazione di Myhill-Nerode R_L, le classi che corrispondono agli elementi di L.

Infatti sia $x \in L$ e supponiamo xR_Ly.

Applicando la definizione con $z = \epsilon$ otteniamo $y \in L$.

Sia $L = L(a^*ba^*)$. Allora xR_Ly se e solo se vale uno dei seguenti casi

La relazione R_L

Sia $L = L(a^*ba^*)$. Allora xR_Ly se e solo se vale uno dei seguenti casi

1. x e y non hanno occorrenze della lettera b

Sia $L = L(a^*ba^*)$. Allora xR_Ly se e solo se vale uno dei seguenti casi

1. x e y non hanno occorrenze della lettera b
2. x e y hanno una sola occorrenza della lettera b

Sia $L = L(a^*ba^*)$. Allora xR_Ly se e solo se vale uno dei seguenti casi

1. x e y non hanno occorrenze della lettera b
2. x e y hanno una sola occorrenza della lettera b
3. x e y hanno più di una occorrenza della lettera b

Sia $L = L(a^*ba^*)$. Allora xR_Ly se e solo se vale uno dei seguenti casi

1. x e y non hanno occorrenze della lettera b
2. x e y hanno una sola occorrenza della lettera b
3. x e y hanno più di una occorrenza della lettera b

Abbiamo quindi tre classi di equivalenza di R_L, che corrispondono ai linguaggi:

$$a^*, \quad a^*ba^*, \quad a^*ba^*b(a \cup b)^*$$

Sia $L = L(a^* b a^*)$.

Sia $L = L(a^*ba^*)$.

Per vedere perché R_L ha tre classi di equivalenza che corrispondono ai linguaggi

$$a^*, \quad a^*ba^*, \quad a^*ba^*b(a \cup b)^*$$

osserviamo che questi tre linguaggi costituiscono una partizione di $\{a, b\}^*$.

Sia $L = L(a^*ba^*)$.

Per vedere perché R_L ha tre classi di equivalenza che corrispondono ai linguaggi

$$a^*, \quad a^*ba^*, \quad a^*ba^*b(a \cup b)^*$$

osserviamo che questi tre linguaggi costituiscono una partizione di $\{a, b\}^*$.

Quindi, presa una stringa in $\{a, b\}^*$, distinguiamo tre casi:
- la stringa non ha occorrenze della lettera b, quindi è della forma a^i, $i \geq 0$;
- la stringa ha una sola occorrenza della lettera b, quindi è della forma a^iba^j, $i, j \geq 0$;
- la stringa ha più di una occorrenza della lettera b, quindi è della forma a^iba^jbz', con $i, j \geq 0, z' \in \{a, b\}^*$.

Sia $L = L(a^* b a^*)$.

Sia $L = L(a^* b a^*)$.

- Se $x = a^i$ e $y = a^h b a^k$, $i, h, k \geq 0$, allora x e y non sono equivalenti perché $x\epsilon \notin L$ mentre $y\epsilon \in L$.

Sia $L = L(a^* ba^*)$.

- Se $x = a^i$ e $y = a^h ba^k$, $i, h, k \geq 0$, allora x e y non sono equivalenti perché $x\epsilon \notin L$ mentre $y\epsilon \in L$.
- Se $x = a^i$ e $y = a^h ba^k bz'$, $i, h, k \geq 0, z' \in \{a, b\}^*$, allora x e y non sono equivalenti perché $xb \in L$ mentre $yb \notin L$.

Sia $L = L(a^* ba^*)$.

- Se $x = a^i$ e $y = a^h ba^k$, $i, h, k \geq 0$, allora x e y non sono equivalenti perché $x\epsilon \notin L$ mentre $y\epsilon \in L$.
- Se $x = a^i$ e $y = a^h ba^k bz'$, $i, h, k \geq 0, z' \in \{a, b\}^*$, allora x e y non sono equivalenti perché $xb \in L$ mentre $yb \notin L$.
- Se $x = a^i$ e $y = a^j$, $i, j \geq 0$, allora x e y sono equivalenti perché $xz \in L$ se e solo se $z = a^h ba^k$, $h, k \geq 0$, e lo stesso è vero per y. Cioè $yz \in L$ se e solo se z ha una sola occorrenza della lettera b.

Sia $L = L(a^*ba^*)$.

- Se $x = a^i$ e $y = a^h ba^k$, $i, h, k \geq 0$, allora x e y non sono equivalenti perché $x\epsilon \notin L$ mentre $y\epsilon \in L$.
- Se $x = a^i$ e $y = a^h ba^k bz'$, $i, h, k \geq 0, z' \in \{a, b\}^*$, allora x e y non sono equivalenti perché $xb \in L$ mentre $yb \notin L$.
- Se $x = a^i$ e $y = a^j$, $i, j \geq 0$, allora x e y sono equivalenti perché $xz \in L$ se e solo se $z = a^h ba^k$, $h, k \geq 0$, e lo stesso è vero per y. Cioè $yz \in L$ se e solo se z ha una sola occorrenza della lettera b.

Un ragionamento analogo si applica negli altri due casi, cioè se $x = a^i ba^j$ oppure se $x = a^i ba^j bz'$, $i, j \geq 0, z' \in \{a, b\}^*$.

Proposizione

Sia $L \subseteq \Sigma^*$, sia E una relazione di equivalenza su Σ^* invariante a destra e tale che L è unione di alcune classi di equivalenza di E. Allora E è un raffinamento di R_L, la relazione di equivalenza di Myhill-Nerode.

Proposizione

Sia $L \subseteq \Sigma^*$, sia E una relazione di equivalenza su Σ^* invariante a destra e tale che L è unione di alcune classi di equivalenza di E. Allora E è un raffinamento di R_L, la relazione di equivalenza di Myhill-Nerode.

Prova. Siano $x, y \in \Sigma^*$. Supponiamo xEy e proviamo xR_Ly.

Proposizione

Sia $L \subseteq \Sigma^*$, sia E una relazione di equivalenza su Σ^* invariante a destra e tale che L è unione di alcune classi di equivalenza di E. Allora E è un raffinamento di R_L, la relazione di equivalenza di Myhill-Nerode.

Prova. Siano $x, y \in \Sigma^*$. Supponiamo xEy e proviamo xR_Ly.

Siccome E è invariante a destra e xEy, allora per ogni $z \in \Sigma^*$ abbiamo $xzEyz$.

Proposizione

Sia $L \subseteq \Sigma^*$, sia E una relazione di equivalenza su Σ^* invariante a destra e tale che L è unione di alcune classi di equivalenza di E. Allora E è un raffinamento di R_L, la relazione di equivalenza di Myhill-Nerode.

Prova. Siano $x, y \in \Sigma^*$. Supponiamo xEy e proviamo xR_Ly.

Siccome E è invariante a destra e xEy, allora per ogni $z \in \Sigma^*$ abbiamo $xzEyz$.

Siccome L è unione di classi di E, otteniamo

$$xz \in L \Leftrightarrow yz \in L$$

cioè xR_Ly.

Teorema

Le seguenti affermazioni sono equivalenti

Teorema

Le seguenti affermazioni sono equivalenti

(a) $L \subseteq \Sigma^*$ *è regolare.*

Teorema

Le seguenti affermazioni sono equivalenti

(a) $L \subseteq \Sigma^*$ *è regolare.*

(b) L *è unione di classi di equivalenza di* E*, dove* E *è una relazione di equivalenza invariante a destra di indice finito.*

Teorema

Le seguenti affermazioni sono equivalenti

(a) $L \subseteq \Sigma^*$ *è regolare.*

(b) L *è unione di classi di equivalenza di* E, *dove* E *è una relazione di equivalenza invariante a destra di indice finito.*

(c) R_L *ha indice finito.*

Prova (cenni).

Prova (cenni).

$(a) \Rightarrow (b)$. Dobbiamo dimostrare che se $L \subseteq \Sigma^*$ è regolare allora L è unione di classi di equivalenza di E, dove E è una relazione di equivalenza invariante a destra di indice finito.

Prova (cenni).

$(a) \Rightarrow (b)$. Dobbiamo dimostrare che se $L \subseteq \Sigma^*$ è regolare allora L è unione di classi di equivalenza di E, dove E è una relazione di equivalenza invariante a destra di indice finito.

Se L è regolare, esiste un DFA $\mathcal{A}$ tale che $L = L(\mathcal{A})$. L è unione di classi della relazione $R_{\mathcal{A}}$ ed $R_{\mathcal{A}}$ è una relazione di equivalenza invariante a destra di indice finito.

$(b) \Rightarrow (c)$. Dobbiamo dimostrare che se L è unione di classi di equivalenza di E, dove E è una relazione di equivalenza invariante a destra di indice finito allora R_L ha indice finito.

$(b) \Rightarrow (c)$. Dobbiamo dimostrare che se L è unione di classi di equivalenza di E, dove E è una relazione di equivalenza invariante a destra di indice finito allora R_L ha indice finito.

Ma abbiamo dimostrato che ogni relazione di equivalenza E invariante a destra e tale che L è unione di alcune classi di equivalenza di E è un raffinamento di R_L.

$(b) \Rightarrow (c)$. Dobbiamo dimostrare che se L è unione di classi di equivalenza di E, dove E è una relazione di equivalenza invariante a destra di indice finito allora R_L ha indice finito.

Ma abbiamo dimostrato che ogni relazione di equivalenza E invariante a destra e tale che L è unione di alcune classi di equivalenza di E è un raffinamento di R_L.

Quindi E è un raffinamento di R_L. Di conseguenza, R_L non può avere più classi di E. Siccome E ha indice finito, cioè ha un numero finito di classi, anche R_L ha indice finito e l'indice di R_L è minore o uguale dell'indice di E.

$(c) \Rightarrow (a)$. Dobbiamo dimostrare che se R_L ha indice finito allora L è regolare.

$(c) \Rightarrow (a)$. Dobbiamo dimostrare che se R_L ha indice finito allora L è regolare.

Denotiamo $[x]$ la classe di $x \in \Sigma^*$ rispetto all'equivalenza R_L. Sia $M' = (Q', \Sigma, \delta', q'_0, F')$ dove

$$\begin{aligned} Q' &= \{[x] \mid x \in \Sigma^*\} \\ q'_0 &= [\epsilon] \\ F' &= \{[x] \mid x \in L\} \\ \delta'([x], a) &= [xa], \quad \text{per ogni } [x] \in Q', a \in \Sigma. \end{aligned}$$

$(c) \Rightarrow (a)$. Dobbiamo dimostrare che se R_L ha indice finito allora L è regolare.

Denotiamo $[x]$ la classe di $x \in \Sigma^*$ rispetto all'equivalenza R_L. Sia $M' = (Q', \Sigma, \delta', q'_0, F')$ dove

$$\begin{array}{rcl} Q' & = & \{[x] \mid x \in \Sigma^*\} \\ q'_0 & = & [\epsilon] \\ F' & = & \{[x] \mid x \in L\} \\ \delta'([x], a) & = & [xa], \quad \text{per ogni } [x] \in Q', a \in \Sigma. \end{array}$$

Q' è un insieme finito e vogliamo concludere che M' è un DFA che riconosce L.

Dobbiamo prima provare che la definizione di δ' è ben posta, cioè che la definizione di δ' non dipende dal rappresentante scelto.

Dobbiamo prima provare che la definizione di δ' è ben posta, cioè che la definizione di δ' non dipende dal rappresentante scelto.

Quindi, siano $x, y \in \Sigma^*$ con $y \in [x]$, cioè $[x] = [y]$. Dobbiamo provare che $\delta'([x], a) = [xa] = [ya] = \delta'([y], a)$.

Dobbiamo prima provare che la definizione di δ' è ben posta, cioè che la definizione di δ' non dipende dal rappresentante scelto.

Quindi, siano $x, y \in \Sigma^*$ con $y \in [x]$, cioè $[x] = [y]$. Dobbiamo provare che $\delta'([x], a) = [xa] = [ya] = \delta'([y], a)$.

Infatti, siccome $y \in [x]$, allora $xR_L y$.

Dobbiamo prima provare che la definizione di δ' è ben posta, cioè che la definizione di δ' non dipende dal rappresentante scelto.

Quindi, siano $x, y \in \Sigma^*$ con $y \in [x]$, cioè $[x] = [y]$. Dobbiamo provare che $\delta'([x], a) = [xa] = [ya] = \delta'([y], a)$.

Infatti, siccome $y \in [x]$, allora $x R_L y$.

Ma R_L è invariante a destra, quindi per ogni $a \in \Sigma$, $xa \ R_L \ ya$ cioè $[xa] = [ya]$.

Dobbiamo prima provare che la definizione di δ' è ben posta, cioè che la definizione di δ' non dipende dal rappresentante scelto.

Quindi, siano $x, y \in \Sigma^*$ con $y \in [x]$, cioè $[x] = [y]$. Dobbiamo provare che $\delta'([x], a) = [xa] = [ya] = \delta'([y], a)$.

Infatti, siccome $y \in [x]$, allora $xR_L y$.

Ma R_L è invariante a destra, quindi per ogni $a \in \Sigma$, $xa\ R_L\ ya$ cioè $[xa] = [ya]$.

Infine, per induzione su $|y|$ si può provare che $\hat{\delta'}([\epsilon], y) = [y]$, per ogni $y \in \Sigma^*$.

Dobbiamo prima provare che la definizione di δ' è ben posta, cioè che la definizione di δ' non dipende dal rappresentante scelto.

Quindi, siano $x, y \in \Sigma^*$ con $y \in [x]$, cioè $[x] = [y]$. Dobbiamo provare che $\delta'([x], a) = [xa] = [ya] = \delta'([y], a)$.

Infatti, siccome $y \in [x]$, allora $x R_L y$.

Ma R_L è invariante a destra, quindi per ogni $a \in \Sigma$, $xa \; R_L \; ya$ cioè $[xa] = [ya]$.

Infine, per induzione su $|y|$ si può provare che $\hat{\delta'}([\epsilon], y) = [y]$, per ogni $y \in \Sigma^*$.

Da questo segue $L(M') = L$. (Ricorda che L è unione di classi della relazione R_L.)

L'automa M' della prova, costruito con le classi di equivalenza di R_L, è l'automa minimale di L.

L'automa M' della prova, costruito con le classi di equivalenza di R_L, è l'automa minimale di L.

Teorema

L'automa $M' = (Q', \Sigma, \delta', q'_0, F')$ è l'unico automa minimale per L, a meno di una ridenominazione degli stati.

Prova (cenni). Se $M = (Q, \Sigma, \delta, q_0, F)$ è un qualsiasi automa che riconosce L, allora R_M è un raffinamento di R_L. Sappiamo che l'indice di R_M è al più $|Q|$. Quindi

$$|Q| \geq \text{ indice di } R_M \geq \text{ indice di } R_L = |Q'|$$

Prova (cenni). Se $M = (Q, \Sigma, \delta, q_0, F)$ è un qualsiasi automa che riconosce L, allora R_M è un raffinamento di R_L. Sappiamo che l'indice di R_M è al più $|Q|$. Quindi

$$|Q| \geq \text{ indice di } R_M \geq \text{ indice di } R_L = |Q'|$$

Ogni DFA che riconosce L ha un numero di stati maggiore o uguale al numero degli stati di M'.

Prova (cenni). Se $M = (Q, \Sigma, \delta, q_0, F)$ è un qualsiasi automa che riconosce L, allora R_M è un raffinamento di R_L. Sappiamo che l'indice di R_M è al più $|Q|$. Quindi

$$|Q| \geq \text{ indice di } R_M \geq \text{ indice di } R_L = |Q'|$$

Ogni DFA che riconosce L ha un numero di stati maggiore o uguale al numero degli stati di M'.

Supponiamo che M ed M' abbiano lo stesso numero di stati. Allora le precedenti diseguaglianze diventano uguaglianze:

$$|Q| = \text{ indice di } R_M = \text{ indice di } R_L = |Q'|$$

Prova (cenni). Se $M = (Q, \Sigma, \delta, q_0, F)$ è un qualsiasi automa che riconosce L, allora R_M è un raffinamento di R_L. Sappiamo che l'indice di R_M è al più $|Q|$. Quindi

$$|Q| \geq \text{ indice di } R_M \geq \text{ indice di } R_L = |Q'|$$

Ogni DFA che riconosce L ha un numero di stati maggiore o uguale al numero degli stati di M'.

Supponiamo che M ed M' abbiano lo stesso numero di stati. Allora le precedenti diseguaglianze diventano uguaglianze:

$$|Q| = \text{ indice di } R_M = \text{ indice di } R_L = |Q'|$$

Ogni stato $q \in Q$ di M è accessibile da q_0.

Prova (cenni). Se $M = (Q, \Sigma, \delta, q_0, F)$ è un qualsiasi automa che riconosce L, allora R_M è un raffinamento di R_L. Sappiamo che l'indice di R_M è al più $|Q|$. Quindi

$$|Q| \geq \text{ indice di } R_M \geq \text{ indice di } R_L = |Q'|$$

Ogni DFA che riconosce L ha un numero di stati maggiore o uguale al numero degli stati di M'.

Supponiamo che M ed M' abbiano lo stesso numero di stati. Allora le precedenti diseguaglianze diventano uguaglianze:

$$|Q| = \text{ indice di } R_M = \text{ indice di } R_L = |Q'|$$

Ogni stato $q \in Q$ di M è accessibile da q_0.

Sia x una stringa tale che $\hat{\delta}(q_0, x) = q$. Identificando q ed $[x]$ otteniamo l'isomorfismo tra M ed M'.

Esempio. Sia $L = L(a^*ba^*)$.
Abbiamo tre classi di equivalenza di R_L, che corrispondono ai linguaggi a^*, a^*ba^*, $a^*ba^*b(a \cup b)^*$:

$$q'_0 = [\epsilon], \quad q'_1 = [b], \quad q'_2 = [bb]$$

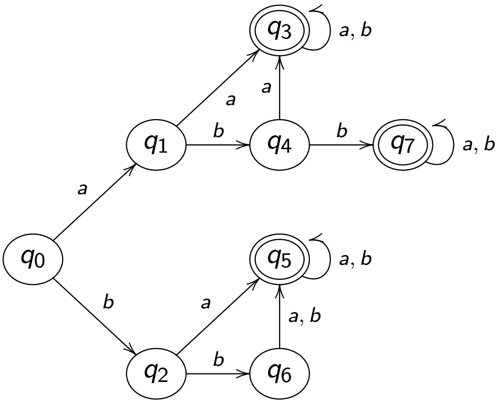
Esempio. Sia $L = L(a^*ba^*)$.
Abbiamo tre classi di equivalenza di R_L, che corrispondono ai linguaggi a^*, a^*ba^*, $a^*ba^*b(a \cup b)^*$:

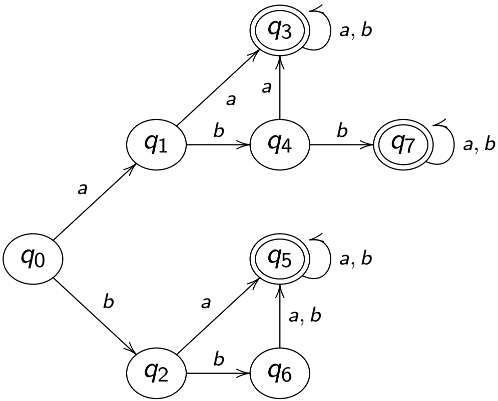
$$q'_0 = [\epsilon], \quad q'_1 = [b], \quad q'_2 = [bb]$$

L'automa minimale di L è $M' = (Q', \Sigma, \delta', q'_0, F')$, dove

$$\begin{array}{rcl} Q' & = & \{q'_0, q'_1, q'_2\} \\ F' & = & \{q'_1\} \\ \delta'(q'_0, a) & = & q'_0, \quad \delta'(q'_0, b) = q'_1, \\ \delta'(q'_1, a) & = & q'_1, \quad \delta'(q'_1, b) = q'_2, \\ \delta'(q'_2, a) & = & q'_2, \quad \delta'(q'_2, b) = q'_2 \end{array}$$

Esempio 2





L'automa M, con stato iniziale q_0, riconosce il linguaggio

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

La relazione R_M è poco interessante: tutti gli stati, tranne quelli finali, definiscono classi con poche parole.

Consideriamo il linguaggio

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

Costruiamo le classi di R_L.

Consideriamo il linguaggio

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

Costruiamo le classi di R_L.

Tutte le stringhe in L formano una sola classe: se $x, y \in L$, per ogni z risulta $xz, yz \in L$. Poniamo $[aa] = D$.

Esempio 2

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

Consideriamo le altre stringhe:

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

Consideriamo le altre stringhe:

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

ϵ non è equivalente alle altre quattro: $\epsilon a \notin L$ ma $ya \in L$ per $y \in L'$, $y \neq \epsilon$. Poniamo $[\epsilon] = A$.

Esempio 2

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a,b\}^*\}.$$

$$\{a,b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

Le stringhe a, b formano una classe. Infatti a, b non sono equivalenti ad ab né a bb.

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

Le stringhe a, b formano una classe. Infatti a, b non sono equivalenti ad ab né a bb.

La lettera b "separa" a e b da ab e bb: $ab \notin L$, $bb \notin L$ mentre $abb \in L$, $bbb \in L$.

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

Le stringhe a, b formano una classe. Infatti a, b non sono equivalenti ad ab né a bb.

La lettera b “separa” a e b da ab e bb: $ab \notin L$, $bb \notin L$ mentre $abb \in L$, $bbb \in L$.

Inoltre a e b sono equivalenti. Infatti $az \in L$ se e solo se $z \in \{aw, baw, bbw \mid w \in \{a, b\}^*\}$ e lo stesso accade per bz. Cioè $bz \in L$ se e solo se $z \in \{aw, baw, bbw \mid w \in \{a, b\}^*\}$.

Esempio 2

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

Le stringhe a, b formano una classe. Infatti a, b non sono equivalenti ad ab né a bb.

La lettera b “separa” a e b da ab e bb: $ab \notin L$, $bb \notin L$ mentre $abb \in L$, $bbb \in L$.

Inoltre a e b sono equivalenti. Infatti $az \in L$ se e solo se $z \in \{aw, baw, bbw \mid w \in \{a, b\}^*\}$ e lo stesso accade per bz. Cioè $bz \in L$ se e solo se $z \in \{aw, baw, bbw \mid w \in \{a, b\}^*\}$.

Poniamo $[a] = B$.

Esempio 2

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

Esempio 2

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

$$L = \{aaw, abaw, abbw, baw, bbaw, bbbw \mid w \in \{a, b\}^*\}.$$

$$\{a, b\}^* \setminus L = \{\epsilon, a, b, ab, bb\} = L'$$

Infine le stringhe ab, bb formano un'altra classe: $abz \in L$ se e solo se $z \in \{aw, bw \mid w \in \{a, b\}^*\}$ e lo stesso accade per $bbz \in L$. Cioè $bbz \in L$ se e solo se $z \in \{aw, bw \mid w \in \{a, b\}^*\}$. Poniamo $[ab] = C$.

Esempio 2

In conclusione le classi di R_L sono quattro:

$$[\epsilon] = A, \quad [a] = B, \quad [ab] = C, \quad [aa] = D$$

In conclusione le classi di R_L sono quattro:

$$[\epsilon] = A, \quad [a] = B, \quad [ab] = C, \quad [aa] = D$$

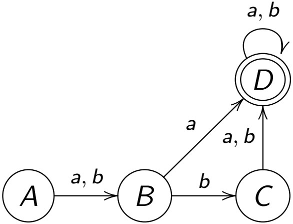
L'automa minimale M' ha quattro stati A, B, C, D, stato iniziale A, stato finale D.

In conclusione le classi di R_L sono quattro:

$$[\epsilon] = A, \quad [a] = B, \quad [ab] = C, \quad [aa] = D$$

L'automa minimale M' ha quattro stati A, B, C, D, stato iniziale A, stato finale D.

Definendo le transizioni come nella prova del teorema di Myhill-Nerode otteniamo lo stesso automa (minimale) costruito mediante l'algoritmo $\mathcal{A}lg$.

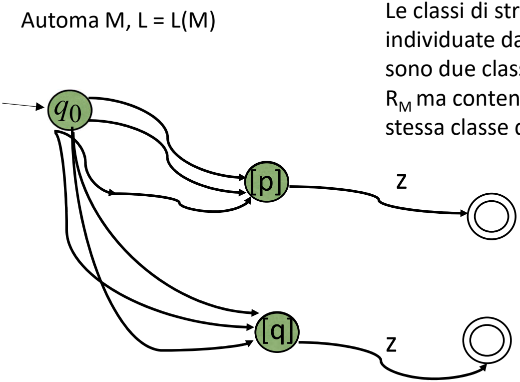


Perché $\mathcal{A}lg$ è corretto, cioè costruisce l'automa minimale per L?

Perché $\mathcal{A}lg$ è corretto, cioè costruisce l'automa minimale per L?

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Perché $\mathcal{A}lg$ è corretto?



Le classi di stringhe individuate da [p] e [q] sono due classi distinte in R_M ma contenute nella stessa classe di R_L

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Se non lo fosse, ci sarebbero due classi di R_M, individuate da due stati p, q di M, contenute in una sola classe di R_L.

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Se non lo fosse, ci sarebbero due classi di R_M, individuate da due stati p, q di M, contenute in una sola classe di R_L.

Quindi, le etichette dei cammini da q_0 a p in M e quelle dei cammini da q_0 a q sarebbero stringhe equivalenti rispetto ad R_L.

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Se non lo fosse, ci sarebbero due classi di R_M, individuate da due stati p, q di M, contenute in una sola classe di R_L.

Quindi, le etichette dei cammini da q_0 a p in M e quelle dei cammini da q_0 a q sarebbero stringhe equivalenti rispetto ad R_L.

Quindi, per ogni stringa z, avremmo che z è etichetta di un cammino da p a uno stato finale se e solo se z è etichetta di un cammino da q a uno stato finale.

Costruzione dell'automa minimale per L

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Se non lo fosse, ci sarebbero due classi di R_M, individuate da due stati p, q di M, contenute in una sola classe di R_L.

Quindi, le etichette dei cammini da q_0 a p in M e quelle dei cammini da q_0 a q sarebbero stringhe equivalenti rispetto ad R_L.

Quindi, per ogni stringa z, avremmo che z è etichetta di un cammino da p a uno stato finale se e solo se z è etichetta di un cammino da q a uno stato finale.

Gli stati p e q sarebbero equivalenti ma questo non è possibile nell'automa M per L ottenuto mediante $\mathcal{A}lg$ (gli stati sono tutti distinguibili).

Perché l'automa M per L ottenuto mediante $\mathcal{A}lg$ è (a meno di una ridenominazione degli stati) lo stesso automa M' costruito mediante R_L?

Se non lo fosse, ci sarebbero due classi di R_M, individuate da due stati p, q di M, contenute in una sola classe di R_L.

Quindi, le etichette dei cammini da q_0 a p in M e quelle dei cammini da q_0 a q sarebbero stringhe equivalenti rispetto ad R_L.

Quindi, per ogni stringa z, avremmo che z è etichetta di un cammino da p a uno stato finale se e solo se z è etichetta di un cammino da q a uno stato finale.

Gli stati p e q sarebbero equivalenti ma questo non è possibile nell'automa M per L ottenuto mediante $\mathcal{A}lg$ (gli stati sono tutti distinguibili).

Formalizziamo questo discorso.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

Consideriamo nuovamente l'automa ottenuto raggruppando stati equivalenti in un solo stato

$$\mathcal{B} = (Q_m, \Sigma, \delta_m, [q_0], F_m)$$

dove

$$\begin{array}{rcl}
Q_m & = & \{[q] \mid q \in Q \text{ e } q \text{ è accessibile da } q_0\} \\
F_m & = & \{[q] \mid q \in F\} \\
\delta_m([q], a) & = & [\delta(q, a)], \quad \text{per ogni } [q] \in Q_m, a \in \Sigma
\end{array}$$

Costruzione dell'automa minimale per L

Sia $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un DFA che riconosce L.

Consideriamo nuovamente l'automa ottenuto raggruppando stati equivalenti in un solo stato

$$\mathcal{B} = (Q_m, \Sigma, \delta_m, [q_0], F_m)$$

dove

$$\begin{array}{rcl} Q_m & = & \{[q] \mid q \in Q \text{ e } q \text{ è accessibile da } q_0\} \\ F_m & = & \{[q] \mid q \in F\} \\ \delta_m([q], a) & = & [\delta(q, a)], \quad \text{per ogni } [q] \in Q_m, a \in \Sigma \end{array}$$

In questo caso $[q]$ denota la classe di q, cioè l'insieme degli stati di $\mathcal{A}$ equivalenti a q.

Perché $\mathcal{B}$ è l'automa minimale di $L = L(\mathcal{A})$?

Perché $\mathcal{B}$ è l'automa minimale di $L = L(\mathcal{A})$?

Supponiamo che non lo sia. Allora $R_{\mathcal{B}}$ ha più classi di R_L, la relazione di equivalenza di Myhill-Nerode di L.

Perché $\mathcal{B}$ è l'automa minimale di $L = L(\mathcal{A})$?

Supponiamo che non lo sia. Allora $R_{\mathcal{B}}$ ha più classi di R_L, la relazione di equivalenza di Myhill-Nerode di L.

Quindi due classi di $R_{\mathcal{B}}$, associate agli stati $[p]$ e $[q]$:

$C_1 = \{x \in \Sigma^* \mid \stackrel{\wedge}{\delta}_m ([q_0], x) = [p]\}$,

$C_2 = \{y \in \Sigma^* \mid \stackrel{\wedge}{\delta}_m ([q_0], y) = [q]\}$

sono contenute nella stessa classe di equivalenza di R_L.

Perché $\mathcal{B}$ è l'automa minimale di $L = L(\mathcal{A})$?

Supponiamo che non lo sia. Allora $R_{\mathcal{B}}$ ha più classi di R_L, la relazione di equivalenza di Myhill-Nerode di L.

Quindi due classi di $R_{\mathcal{B}}$, associate agli stati $[p]$ e $[q]$:

$C_1 = \{x \in \Sigma^* \mid \overset{\wedge}{\delta}_m ([q_0], x) = [p]\}$,
$C_2 = \{y \in \Sigma^* \mid \overset{\wedge}{\delta}_m ([q_0], y) = [q]\}$
sono contenute nella stessa classe di equivalenza di R_L.

Quindi, se $x \in C_1$ e $y \in C_2$, per ogni z, $xz \in L \Leftrightarrow yz \in L$, cioè

$$\overset{\wedge}{\delta}_m ([p], z) \in F \;\Leftrightarrow \overset{\wedge}{\delta}_m ([q], z) \in F$$

il che è assurdo perché p e q sarebbero stati equivalenti e sappiamo che non lo sono.

Consideriamo il problema di decisione:

Dati due DFA $\mathcal{A}$ e $\mathcal{B}$, essi sono equivalenti, cioè $L(\mathcal{A}) = L(\mathcal{B})$?

Consideriamo il problema di decisione:

Dati due DFA $\mathcal{A}$ e $\mathcal{B}$, essi sono equivalenti, cioè $L(\mathcal{A}) = L(\mathcal{B})$?

L'esistenza di algoritmi per la costruzione dell'automa minimale permette di provare la decidibilità del linguaggio associato

$$\{\langle \mathcal{A}, \mathcal{B} \rangle \mid \mathcal{A}, \mathcal{B} \text{ sono DFA e } L(\mathcal{A}) = L(\mathcal{B})\}$$

Teorema

Per $L \subseteq \Sigma^*$ sono equivalenti le condizioni

- *L è riconosciuto da un DFA*
- *L è riconosciuto da un NFA*
- *L è denotato da un'espressione regolare*
- *L è generato da una grammatica regolare*
- *L è unione di classi di un'equivalenza invariante a destra di indice finito.*

Ognuna di queste condizioni definisce i linguaggi regolari.