UNIVERSITÀ
DEGLI STUDI
DI TORINO

Minimizzazione di DFA

a.a. 2018-019

Riferimenti bibliografici

Compilatori: principi, tecniche e strumenti
A.V. Aho, M. S. Lam, R. Sethi, J.D. Ullman

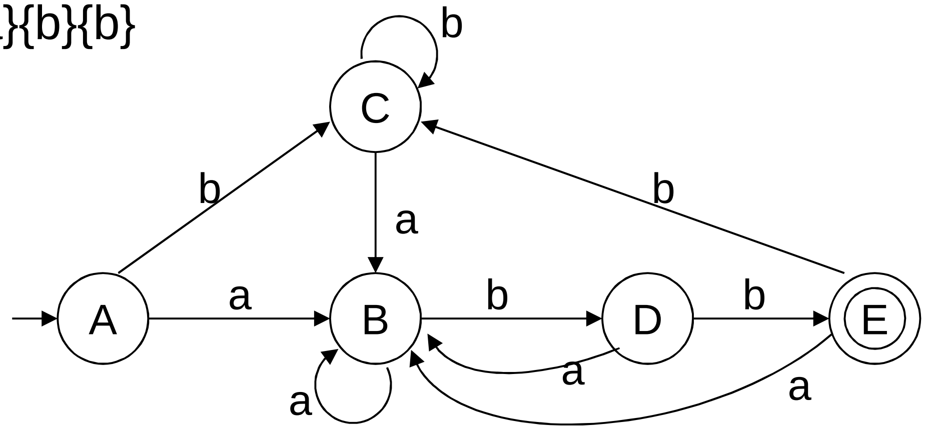
Analisi lessicale [cap.3]
3.9.6 Minimizzazione del numero degli stati di un DFA

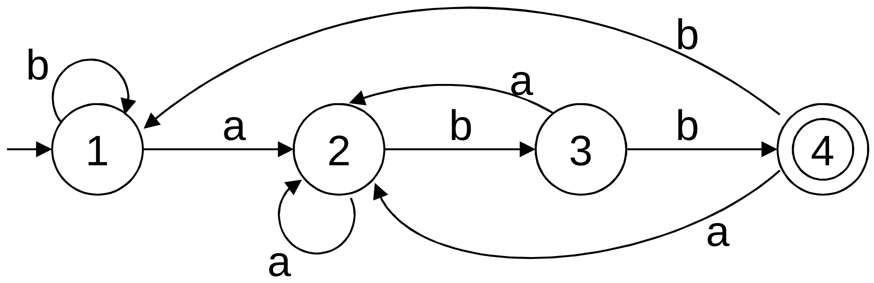
Minimizzazione di automi finiti deterministici

Due automi sono equivalenti se accettano lo stesso linguaggio.

I due automi deterministici seguenti sono equivalenti (entrambi riconoscono il linguaggio (a + b)*abb), ma non uguali.

L = ({a} + {b})*{a}{b}{b}





Uguaglianza di automi finite e automa minimo

Minimizzazione di un automa a stati finiti

Dato un automa a stati finiti *deterministico* D possiamo costruire l'automa a stati finiti "più piccolo" (con il minimo numero di stati) equivalente a D, unico a meno del nome degli stati,

- eliminando gli stati "non raggiungibili dallo stato iniziale" (facile) e
- "fondendo" gli stati "indistinguibili".

Stati indistinguibili

Sia D = $\langle S, \Sigma, \delta, s_0, F \rangle$ un automa finito <u>deterministico</u> senza stati inutili e s e t siano due suoi stati.

1. $x \in \Sigma^*$ *<u>distingue</u>* tra s e t se uno e uno solo tra gli stati $\hat{\delta}(s, x)$ e $\hat{\delta}(t, x)$ è finale.

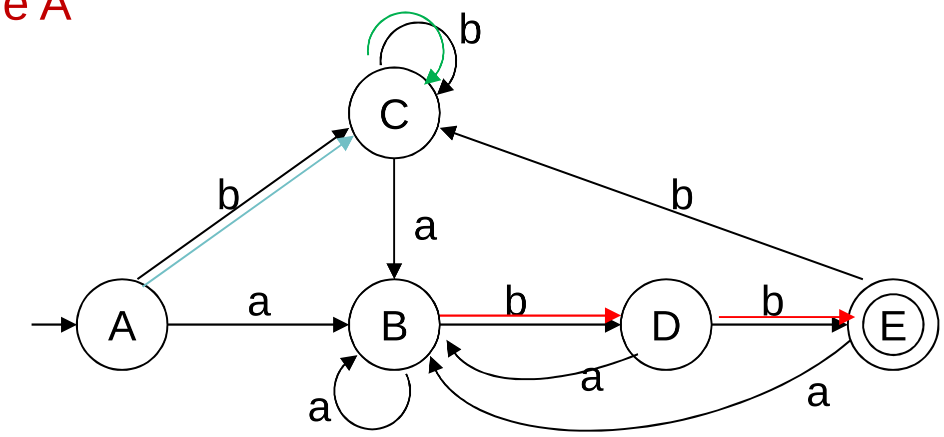
   Ovvero: x distingue s da t se esattamente uno degli stati raggiungibili da s e da t mediante un percorso etichettato con la stringa x è di accettazione).

2. s e t sono k-*<u>indistinguibili</u>* se e solo se nessuna stringa di lunghezza $\leq$ k distingue s da t.

3. s e t sono *<u>indistinguibili</u>* se e solo se nessuna stringa (di qualunque lunghezza) distingue s da t (ovvero sono k-indistinguibili per ogni $k \geq 0$).

ESEMPIO.

bb *distingue* B e A

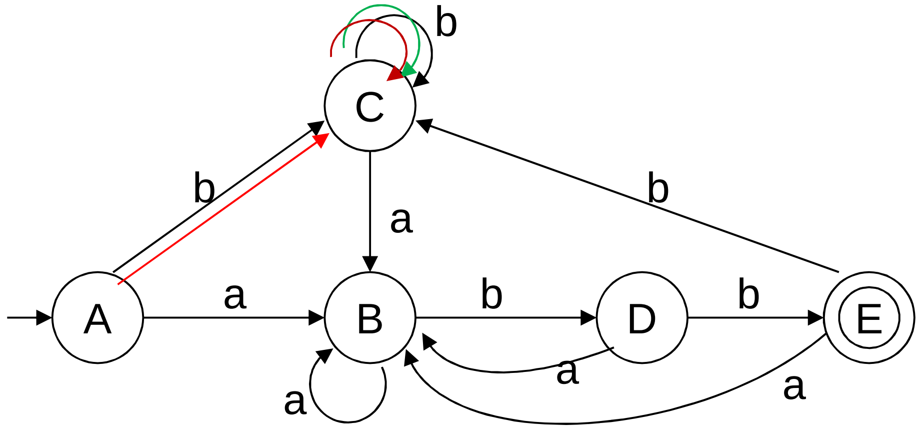


$\hat{\delta}$(B, bb) = E   finale

$\hat{\delta}$(A, bb) = C non finale

Minimizzazione di automi finiti deterministici

Invece A e C non sono distinguibili. Per esempio su bb:



$\hat{\delta}$ (A, bb) = C   non finale

$\hat{\delta}$ (C, bb) = C non finale

Ma attenzione: per dire che A e C non sono distinguibili bisogna essere sicuri che *per tutte* le stringhe x

$\hat{\delta}$ (A, x) finale *se e solo se* $\hat{\delta}$ (C, x) finale

Stati indistinguibili

Possiamo cercare gli stati indistinguibili considerando le stringhe in ordine di lunghezza.

Osservazioni:

1. La stringa vuota ε *distingue* ogni stato finale da ogni stato non finale, per definizione. Gli stati finali (e quelli non finali) sono 0-indistingubili
2. Per esempio, due stati *s* e *t* non sono 1-indistinguibili se, per almeno un simbolo dell'alfabeto *a*, uno e uno solo tra gli stati $\delta(s,a)$ e $\delta(t,a)$ appartiene a F.
3. In generale due stati s e t non sono k+1-indistinguibili solo se per almeno un $a \in \Sigma$ $\delta(s,a)$ e $\delta(t,a)$ non sono k-indistinguibili.
   Infatti se qualche stringa ax di lunghezza k+1 distingue s e t allora x di lunghezza k distingue $\delta(s,a)$ e $\delta(t,a)$.
4. La relazione di k-indistinguibilità (come quella di indistinguibilità) è transitiva. Se s è k-indistinguibile da t e t è k-indistinguibile da r allora s è k-indistinguibile da r.

Partizione degli stati

L'insieme degli stati Q di un automa può essere partizionato in una collezione di sottoinsiemi *disgiunti* $\Pi = (S_1,\ldots, S_n)$ tali che

- $S_1 \cup S_2 \ \ldots. \cup S_n = Q$
- ogni S_i $(1 \leq i \leq n)$ contiene solo stati indistinguibili
- due stati appartenenti a sottoinsiemi diversi sono distinguibili

Per esempio nel caso dell'esempio precedente si ha che

$$\Pi = (\{A, C\}, \{B\}, \{D\}, \{E\})$$

Ma come calcolare la partizione Π?

L'algoritmo di minimizzazione

L'idea dell'algoritmo di minimizzazione è quella di costruire una sequenza di partizioni degli stati:

$$\Pi_0 = (S\text{-}F, F),\ \Pi_1, \Pi_2, \ldots, \Pi_i, \ldots$$

tali che i gruppi della partizione Π_i contengano tutti gli stati *i-indistinguibili*

Le partizioni Π_i possono essere costruite induttivamente patendo da Π_0:

1. Π_0 è semplicemente la partizione che divide gli stati in finali e non finali
2. Ovviamente che se s e t stanno in gruppi distinti in una partizione Π_i staranno in gruppi distinti anche nella partizione Π_{i+1} . Al crescere di i le partizioni possono solo raffinarsi (cioè frammentare i gruppi che le compongono) o restare eguali.
3. Per ogni *i*, due stati s e t stanno nello stesso gruppo della partizione di Π_{i+1} (sono i+1-indistinguibili) se e solo se per ogni $a \in \Sigma$ (alfabeto su cui è definito l'automa) $\delta(s, a)$ e $\delta(t, a)$ stanno nello *stesso* gruppo della partizione Π_i .

L'algoritmo di minimizzazione

Quando fermarsi?

Si può facilmente dimostrare che se $\Pi_i = \Pi_{i+1}$, la partizione degli stati non cambia più al crescere di i, cioè $\Pi_i = \Pi_{i+1} = \Pi_{i+2} = \ldots = \Pi_{i+k} = \ldots$ Allora Π_i (o Π_{i+1}) è la partizione finale Π, che contiene i gruppi degli stati indistinguibili.

Infatti se $\Pi_i = \Pi_{i+1}$ e ci fossero due stati r e s i+1-indistinguibili ma non i+2-indistinguibili ci dovrebbe essere un $a \in \Sigma$ tale che $\delta(s, a)$ e $\delta(t, a)$ non sono i+1-indistinguibili. Ma allora, siccome $\Pi_i = \Pi_{i+1}$ $\delta(s, a)$ e $\delta(t, a)$ non sono i-indistinguibili e quindi s e t non sono i+1-indistinguibili, contro l'ipotesi. Quindi deve essere $\Pi_{i+2} = \Pi_{i+1}$. E così via.

Poiché gli stati sono finiti l'algoritmo deve per forza terminare.

Algoritmo di minimizzazione

1. L'algoritmo parte dalla partizione $\Pi = \Pi_0$ che distingue gli stati finali da quelli non finali e costruisce progressivamente partizioni i cui gruppi sono insiemi di stati non ancora identificati come distinguibili.
2. Due stati appartenenti ad insiemi diversi di una partizione sono invece già stati identificati come distinguibili.
3. Quando la partizione non può più essere modificata spezzando un gruppo in gruppi più piccoli, l'algoritmo costruisce il DFA minimo prendendo come stati i gruppi della partizione stessa ($\Pi = \Pi_{final}$).

1. Il procedimento fondamentale consiste nel considerare un generico gruppo $\{s_1, s_2, \ldots, s_n\}$ e verificare per ogni coppia di stati s_i e s_j, se almeno un simbolo dell'alfabeto di input a li "distingue" nel senso che le transizioni da s_i e s_j con il simbolo a portano in stati che sono in gruppi diversi della partizione precedente.
   Usare dove possibile la transitività.

Algoritmo di minimizzazione

Input: un automa finito deterministico $D = \langle S, \Sigma, \delta, s_0, F \rangle$.
Output: un automa deterministico $D' = \langle S', \Sigma, \delta', s'_0, F' \rangle$ equivalente a D con il minimo numero di stati.

Algoritmo:

1) Sia Π la partizione iniziale costituita dai due gruppi di stati F e S – F.
2) Applicare l'algoritmo per costruire una nuova partizione Π_{new}.
3) Se $\Pi_{new} = \Pi$, allora $\Pi_{final} = \Pi_{new}$ e si procede con il passo (4), altrimenti ripetere il passo (2) con Π_{new} al posto di Π.
4) Costruire l'automa $D' = \langle S', \Sigma, \delta', s'_0, F' \rangle$ nel modo seguente:
   - Per ogni gruppo di Π_{final} scegliere uno stato come rappresentativo del gruppo. S' è l'insieme degli stati rappresentativi dei diversi gruppi.
   - δ': $\delta'(s, a) = t$ se $\delta(s, a) = r$ e r è nel gruppo di cui t è il rappresentante.
   - s'_0 è lo stato che rappresenta il gruppo che contiene lo stato iniziale di D, s_0
   - F' è costituito dai rappresentatnti dei gruppi di stati finali indistinguibili

Algoritmo di minimizzazione

Costruzione della nuova partizione Π_{new}:

Inizializza Π_{new} a Π;
<u>for</u> ogni gruppo G di Π {

suddividi G in sottogruppi tali che due stati *s* e *t* appartengano allo stesso sottogruppo se e solo se per ogni $a \in \Sigma$ gli stati *s* e *t* hanno transizioni per il simbolo *a* verso stati appartenenti allo stesso gruppo di Π;

sostituisci G in Π_{new} con il nuovo insieme di sottogruppi;

}

<u>*Nota*</u>: ogni gruppo contiene tutti stati finali o tutti stati non finali poiché si parte con i due gruppi F e Q-F e il procedimento forma unicamente sottogruppi di gruppi preesistenti.

Automa minimo

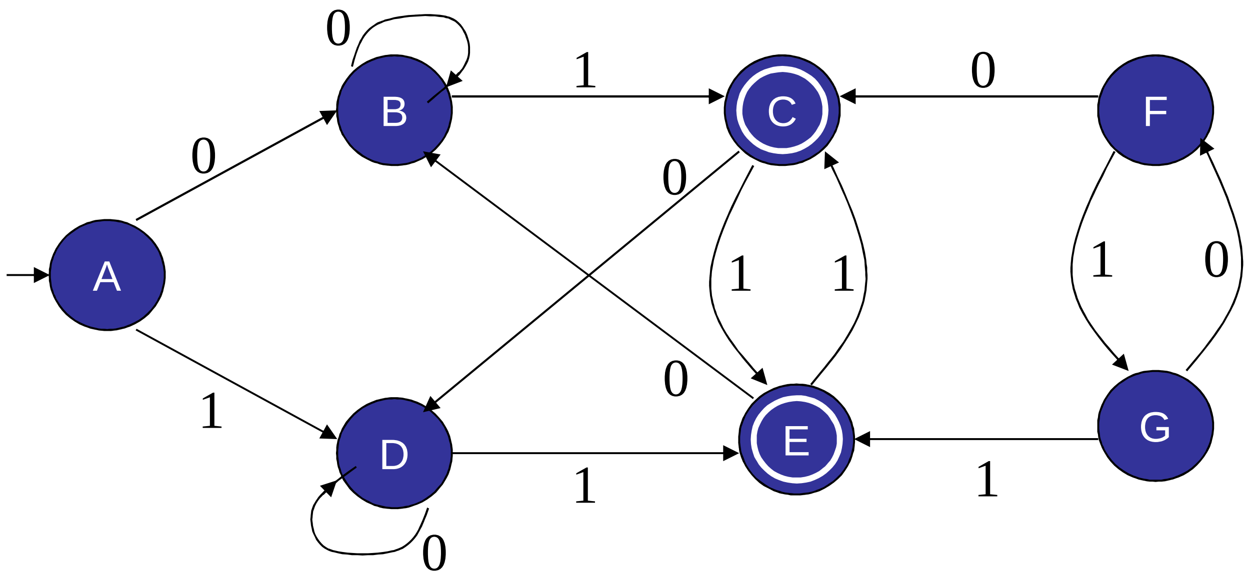
Sia *B* il DFA ottenuto applicando l'algoritmo di minimizzazione al DFA *A*.

Allora:

1. L(*B*) = L(*A*).
2. L'automa *B* è minimo nel senso che non esiste nessun automa equivalente ad *A* con un numero di stati inferiore al numero di stati di B.
3. L'automa minimo è unico, a meno del nome degli stati.

<u>*Nota*</u>: Un modo semplice per verificare se due automi finiti accettano lo stesso linguaggio è quello di minimizzarli. I due automi sono equivalenti se e solo se gli automi minimi ottenuti sono uguali a meno dei nomi (isomorfi).

Esempio



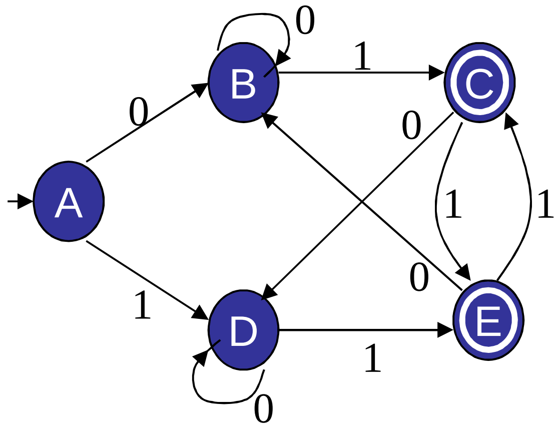
F e G sono stati non raggiungibili dallo stato iniziale e perciò li eliminiamo

Π_0 = {A, B, D}, {C, E}

Π_1 = {A}, {B, D}, {C, E}

Π_2 = {A}, {B, D}, {C, E}

Esempio



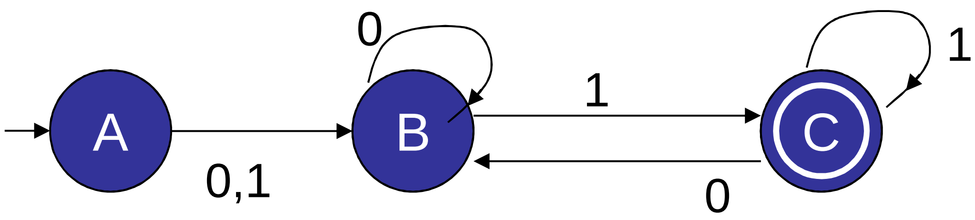
L'automa minimo è:

< {A, B, C}, {0, 1}, δ, A, {C}>

$\delta(A, 0) = B$ $\delta(A, 1) = B$

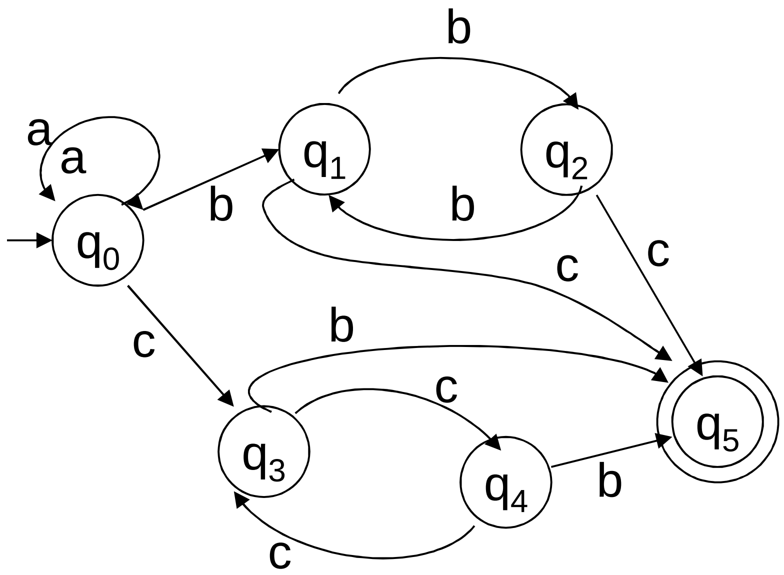
$\delta(B, 0) = B$ $\delta(B, 1) = C$

$\delta(C, 0) = B$ $\delta(C, 1) = C$

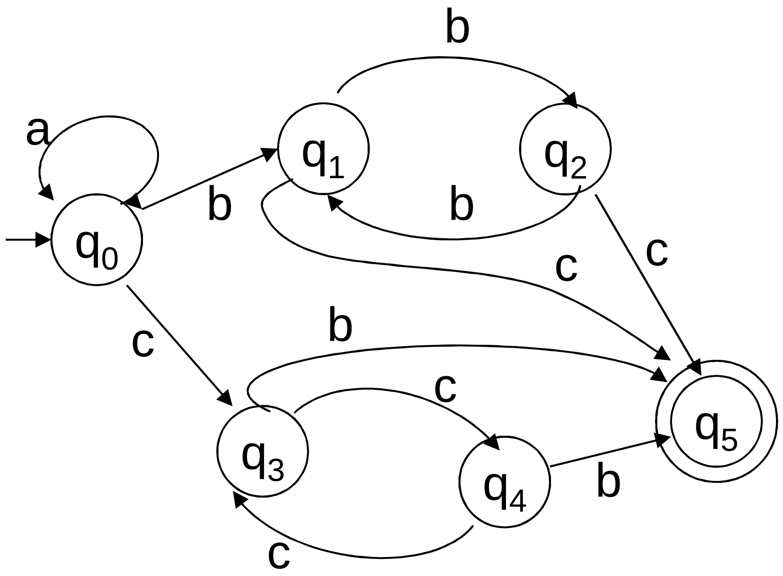


Esempio

$< \{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{\delta(q_0,a) = q_0, \delta(q_0,b) = q_1,$
$\delta(q_0,c) = q_3, \delta(q_1,b) = q_2, \delta(q_2,b) = q_1, \delta(q_1,c) = q_5, \delta(q_2,c\ ) = q_5,$
$\delta(q_3,c) = q_4, \delta(q_4,c) = q_3, \delta(q_3,b) = q_5, \delta(q_4,b) = q_5\}, q_0, \{q_5\} >$



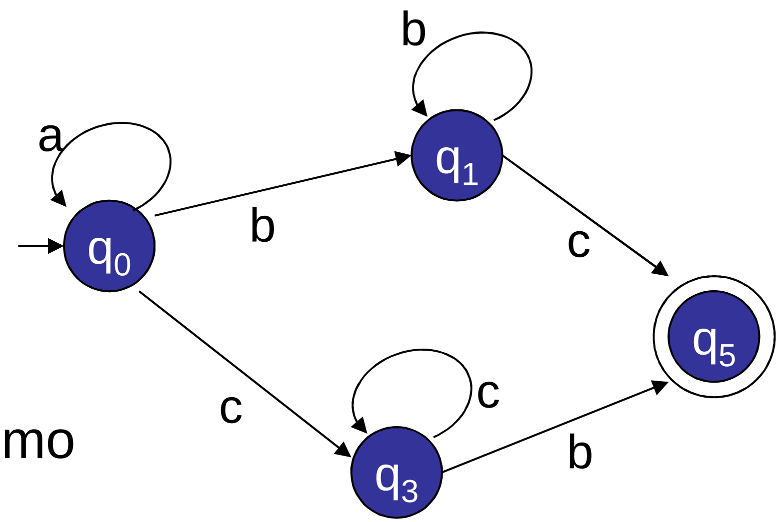
Esempio



$\Pi_0 = \{q_0, q_1, q_2, q_3, q_4\}, \{q_5\}$

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

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



Automa minimo

Esercizi

Per ognuno dei seguenti automi costruire l'automa minimo equivalente;