

Linguaggi Formali e Traduttori

2.1 Automi a stati finiti deterministici (DFA).

- Analisi lessicale
- Descrivere token (di Java)
- Esempio
- Automa a stati finiti
- Un semplice esempio di riconoscimento
- Un semplice esempio di riconoscimento
- La soluzione come automa a stati finiti
- Automi a stati finiti
- Linguaggio riconosciuto da un DFA
- Tabelle di transizione
- Esempio: numeri binari pari
- Esempio: esiste a seguita da bb
- Esempio: ogni a è seguita da bb
- Esempio: numeri binari multipli di 5
- Esercizi sulla definizione di DFA
- Esercizi sulla comprensione di DFA

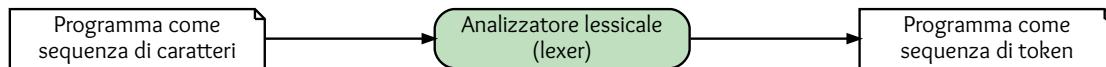
È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Analisi lessicale

Scopo dell'analisi lessicale

Riconoscere **sequenze di caratteri** che rappresentano elementi atomici del programma

Tali sequenze di caratteri sono dette **token** o **lessemi**



Esempi di token

- costanti (42, 1.5, true, "questa è una stringa", ...)
- identificatori (i, metodo, String, ...)
- parole chiave (class, public, for, if, ...)
- operatori (<=, +, ==, ...)
- simboli di punteggiatura (,, ;, (,), {}, ...)
- ...

Descrivere token (di Java)

Costante numerica intera

Una sequenza non vuota di cifre decimali, eventualmente preceduta da + o -

- 0
- -42

Costante numerica con virgola

Due sequenze (di cui almeno una non vuota) di cifre decimali separate da .

- 0.5
- .5

Identificatore

Una sequenza non vuota di lettere, numeri e _ che non inizia con un numero e che contiene almeno un carattere diverso da _

- _i
- metodo

Esempio

Sequenza di caratteri

```
public static int metodo(int n) {  
    int r = 1;  
    for (int i = 1; i <= n; i++)  
        r = r * i;  
    return r;  
}
```

Sequenza di token

- parola chiave public
- parola chiave static
- identificatore int
- identificatore metodo
- parentesi aperta (
- ...

Nota

- alcune sequenze di caratteri (spazi/commenti) vengono **scartate** dal lexer, ma possono essere fondamentali per **separare token adiacenti** (es. int e metodo)

Automa a stati finiti

Che cos'è

- automa = macchina che **riconosce** stringhe
- stati finiti = con **memoria finita** (l'automa ricorda “poche” cose della stringa)
- input dell'automa = una stringa
- output dell'automa = **sì** se la stringa è riconosciuta, **no** altrimenti

Come funziona

- l'automa legge la stringa **un simbolo alla volta**, da sinistra verso destra
 - ⇒ l'automa ha una visione locale e limitata
- ogni simbolo letto altera lo stato dell'automa
 - ⇒ l'automa “ricorda” le caratteristiche della stringa letta usando lo stato
- quando la stringa è stata letta interamente, l'automa risponde “sì” se si trova in un cosiddetto **stato finale** e “no” altrimenti

Un semplice esempio di riconoscimento

Problema

Sono in una stanza con un recipiente contenente un numero impreciso (ma all'apparenza molto grande) di biglie. A parte il recipiente, nella stanza c'è solo un interruttore che accende e spegne una lampada. Ho poca memoria. Ho il compito di svuotare il recipiente e dire "sì" se il numero di biglie è dispari, "no" altrimenti.

Soluzione (difettosa)

- Uso un **contatore** per ricordare il numero di biglie, inizialmente posto a 0
- Estraggo le biglie una alla volta, a ogni estrazione incremento il contatore di 1
- Quando il recipiente è vuoto, dico "sì" se il valore del contatore è dispari, "no" altrimenti

Un semplice esempio di riconoscimento

Problema

Sono in una stanza con un recipiente contenente un numero impreciso (ma all'apparenza molto grande) di biglie. A parte il recipiente, nella stanza c'è solo un interruttore che accende e spegne una lampada. Ho poca memoria. Ho il compito di svuotare il recipiente e dire "sì" se il numero di biglie è dispari, "no" altrimenti.

Soluzione (difettosa)

- Uso un **contatore** per ricordare il numero di biglie, inizialmente posto a 0
- Estraggo le biglie una alla volta, a ogni estrazione incremento il contatore di 1
- Quando il recipiente è vuoto, dico "sì" se il valore del contatore è dispari, "no" altrimenti

Cosa non va in questa soluzione

- La quantità di informazione da ricordare dipende dal numero di biglie nel recipiente
- Il contatore può diventare arbitrariamente grande (= non è "a stati finiti")

Un semplice esempio di riconoscimento

Problema

Sono in una stanza con un recipiente contenente un numero imprecisato (ma all'apparenza molto grande) di biglie. A parte il recipiente, nella stanza c'è solo un interruttore che accende e spegne una lampada. Ho poca memoria. Ho il compito di svuotare il recipiente e dire "sì" se il numero di biglie è dispari, "no" altrimenti.

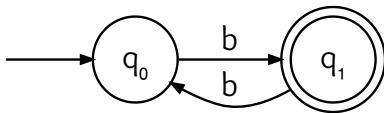
Osservazione

- I numeri pari e dispari si alternano

Soluzione

- Uso la lampada per ricordare se il numero di biglie estratte è pari (lampada spenta) o dispari (lampada accesa)
- Mi assicuro che la lampada sia inizialmente spenta (inizialmente ho estratto 0 biglie e 0 è un numero pari)
- Estraggo le biglie una alla volta, a ogni estrazione premo l'interruttore
- Quando il recipiente è vuoto, dico "sì" se la lampada è accesa e "no" altrimenti

La soluzione come automa a stati finiti



- ogni cerchio rappresenta uno **stato** dell'automa (q_0 = lampada spenta, q_1 = lampada accesa)
- gli archi etichettati “b” rappresentano le **transizioni di stato**, ovvero come cambia lo stato dell’automa (e della lampada) a ogni estrazione di biglia
- la freccia entrante in q_0 indica che q_0 è lo stato **iniziale** (all’inizio la lampada è spenta)
- il doppio cerchio attorno a q_1 indica che q_1 è lo stato **finale o di accettazione** (se l’automa si trova in questo stato quando le biglie sono finite, allora il loro numero era dispari)

Note

- in generale, possono esserci zero o più stati finali
- in generale, transizioni diverse possono essere etichettate con simboli diversi

Automi a stati finiti

Definizione

Un **automa a stati finiti** (detto anche **DFA**, da Deterministic Finite-state Automaton) è una quintupla $A = (Q, \Sigma, \delta, q_0, F)$ dove:

- Q è un insieme finito di **stati**
- Σ è l'**alfabeto** riconosciuto dall'automa
- $\delta : Q \times \Sigma \rightarrow Q$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'insieme di **stati finali**

Esempio

Per l'automa della [slide 8](#) abbiamo:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{b\}$
- $\delta = \{(q_0, b, q_1), (q_1, b, q_0)\}$
- $F = \{q_1\}$

Linguaggio riconosciuto da un DFA

Definizione

La **funzione di transizione estesa** dell'automa $A = (Q, \Sigma, \delta, q_0, F)$ è la funzione $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ definita per induzione sul suo secondo argomento come segue:

$$\hat{\delta}(q, \varepsilon) = q \quad \hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$$

Definizione

Il **linguaggio riconosciuto** (o **accettato**) dall'automa $A = (Q, \Sigma, \delta, q_0, F)$ è denotato da $L(A)$ e definito come segue:

$$L(A) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

Esempio

Per l'automa in [slide 8](#) abbiamo $L(A) = \{b^{2n+1} \mid n \in \mathbb{N}\}$

Definizione

Un linguaggio L si dice **regolare** se esiste un automa A tale che $L = L(A)$.

Tabelle di transizione

Un DFA si può rappresentare comodamente anche in forma tabellare. Per esempio, per l'automa visto in [slide 8](#) avremo:

Stato	b
→ q ₀	q ₁
* q ₁	q ₀

- Le **righe** corrispondono agli **stati** dell'automa
- Le **colonne** corrispondono ai **simboli** dell'alfabeto dell'automa
- Lo **stato iniziale** è marcato con →
- Ogni **stato finale** è marcato con *
- La **cella** corrispondente alla riga **q** e alla colonna **a** contiene $\delta(q, a)$

Esempio: numeri binari pari

Progettare un automa che riconosce le stringhe di bit che rappresentano **numeri pari**

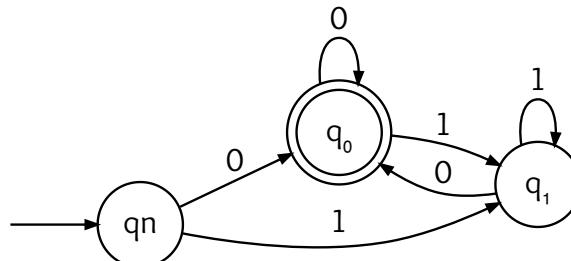
Osservazione

- le stringhe devono essere **non vuote** e **terminare con 0**

Stati

- q_n = non ho ancora riconosciuto alcun bit
- q_0 = ho riconosciuto almeno un bit, l'ultimo era 0
- q_1 = ho riconosciuto almeno un bit, l'ultimo era 1

Soluzione



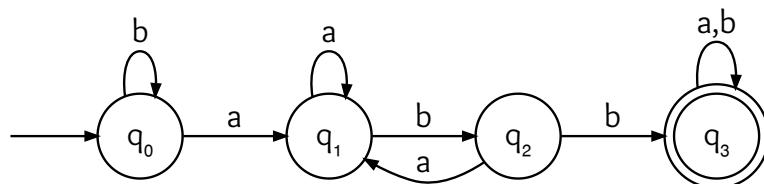
Esempio: esiste a seguita da bb

Definire un automa che riconosce le stringhe di a e b in cui **esiste** una a seguita da bb

Stati

- q_0 = tutti i simboli riconosciuti fino ad ora erano b
- q_1 = l'ultimo simbolo riconosciuto era una a
- q_2 = gli ultimi due simboli riconosciuti erano ab
- q_3 = ho riconosciuto una a seguita da bb

Soluzione



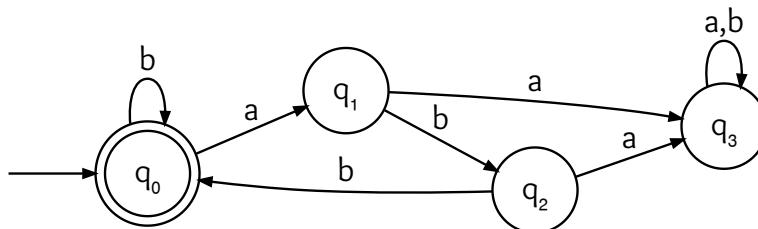
Esempio: ogni a è seguita da bb

Definire un automa che riconosce le stringhe di a e b in cui **ogni** a è seguita da bb

Stati

- q_0 = ogni a riconosciuta era seguita da bb
- q_1 = l'ultimo simbolo riconosciuto era una a
- q_2 = gli ultimi due simboli riconosciuti erano ab
- q_3 = ho riconosciuto una a seguita da una sequenza diversa da bb

Soluzione



Nota

Una volta raggiunto lo stato q_3 l'automa non ha più speranze di riconoscere alcuna stringa. Per tale motivo, q_3 è detto **stato pozzo**

Esempio: numeri binari multipli di 5

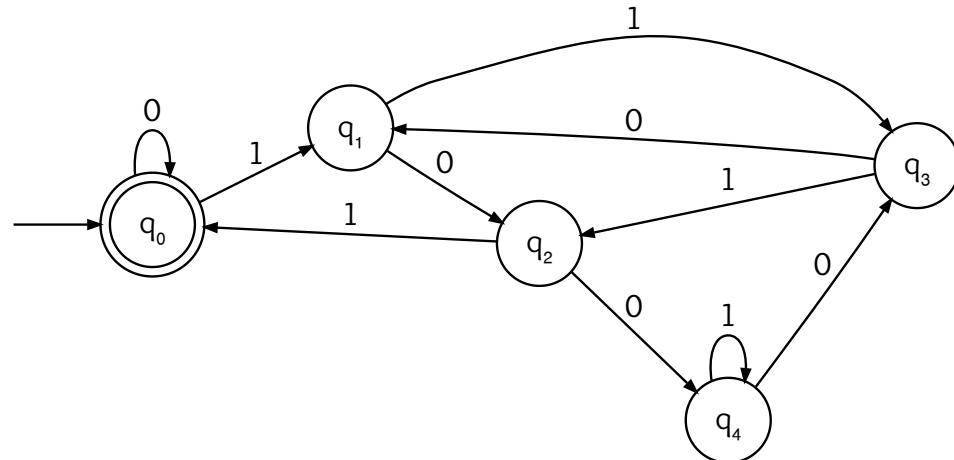
Definire un automa che riconosce le stringhe di bit che rappresentano **multipli di 5**

Osservazioni

Possiamo calcolare il resto della divisione per 5 del numero riconosciuto sapendo che:

- $(2n) \bmod 5 = 2(n \bmod 5) \bmod 5$
- $(2n + 1) \bmod 5 = (2(n \bmod 5) + 1) \bmod 5$

Soluzione

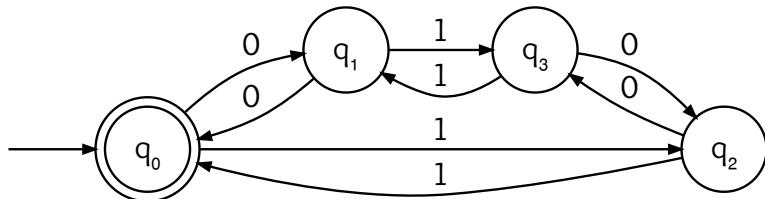
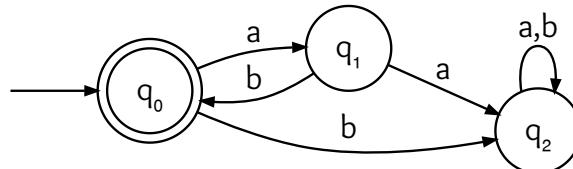
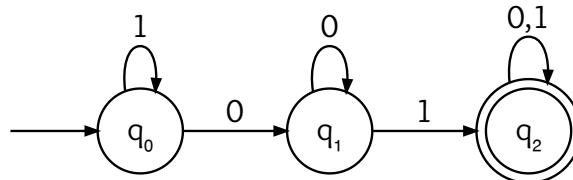


Esercizi sulla definizione di DFA

1. Determinare la rappresentazione tabellare di tutti gli automi visti a lezione.
2. Definire un DFA sull'alfabeto { a, c, i, o } che riconosca la sola stringa ciao.
3. Definire un DFA sull'alfabeto { a, c, i, o } che riconosca le stringhe che contengono al proprio interno la sottostringa ciao, eventualmente preceduta e/o seguita da altri simboli.
4. Definire un DFA sull'alfabeto { a, c, i, o } che riconosca le stringhe che contengono al proprio interno i simboli c, i, a e o in quest'ordine, ciascuno eventualmente preceduto e/o seguito da altri simboli.
5. Definire un DFA sull'alfabeto { 0, 1 } che riconosca le stringhe di lunghezza arbitraria che iniziano con due 0 e terminano con due 1.
6. Definire un DFA sull'alfabeto { a, b, c } che riconosca le stringhe in cui sono presenti almeno due simboli uguali consecutivi.
7. Definire un DFA sull'alfabeto { a, b, c } che riconosca le stringhe in cui **non** sono presenti due simboli uguali consecutivi. Che relazione c'è tra questo automa e quello dell'esercizio precedente?
8. Definire un DFA sull'alfabeto { a, b, c } che riconosca le stringhe ordinate, assumendo $a \leq b \leq c$.
9. Definire un DFA sull'alfabeto { /, *, c } che riconosca le stringhe che iniziano con /*, che finiscono con */ e in cui non ci siano altre occorrenze di * seguite da /.
10. Trovare un DFA più semplice (con meno stati e transizioni) ma equivalente (che riconosce lo stesso linguaggio) di quello in [slide 12](#).

Esercizi sulla comprensione di DFA

Descrivere a parole il linguaggio riconosciuto dai seguenti automi:



Linguaggi Formali e Traduttori

2.2 Automi a stati finiti non deterministici (NFA).

- Sommario
- Esempio di riconoscimento non deterministico
- La soluzione come automa non deterministico
- Automi a stati finiti **non deterministici**
- Linguaggio riconosciuto da un NFA
- Rappresentazione tabellare di NFA
- DFA → NFA
- NFA → DFA
- NFA → DFA: costruzione per sottoinsiemi
- Esempio: stringhe che terminano con abb
- Esempio: ogni a è seguita da bb
- Esempio: stringhe che terminano con abb
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Automa deterministico

Automa in cui la transizione di stato è **univocamente determinata** dallo stato corrente e dal prossimo simbolo nella stringa da riconoscere

$$\delta : Q \times \Sigma \rightarrow Q$$

Automa non deterministico

Automa che può “scegliere” transizioni diverse a parità di stato corrente e prossimo simbolo nella stringa da riconoscere

$$\delta : Q \times \Sigma \rightarrow \wp(Q)$$

In questa lezione

1. Introduciamo la classe degli automi a stati finiti **non deterministici**
2. Mostriamo che ogni linguaggio riconosciuto da un automa non deterministico può essere riconosciuto anche da un automa deterministico il quale, però, può avere più stati e/o transizioni di quello non deterministico

Esempio di riconoscimento non deterministico

Problema

Sono in una stanza con un recipiente contenente un numero imprecisato (ma all'apparenza molto grande) di biglie. Ho il compito di svuotare il recipiente e dire “sì” se il numero di biglie è dispari, “no” altrimenti. **Non c'è la lampada!**

Osservazioni

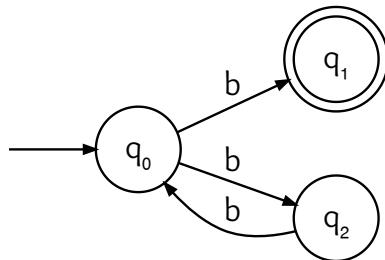
- per ogni $n \geq 2$, il numero $n - 2$ è pari se e solo se n è pari
- per ogni $n \geq 2$, il numero $n - 2$ è dispari se e solo se n è dispari

Soluzione

Mi comporto diversamente in base a quante biglie vedo nel recipiente:

- Se il recipiente è vuoto, dico “no”
- Se il recipiente contiene **una sola biglia**, la rimuovo e dico “sì”
- Se il recipiente contiene **due o più biglie**, ne rimuovo due e ripeto

La soluzione come automa non deterministico



- **q_0** = guardo il recipiente e decido cosa fare, se non ci sono più biglie dico “no”
- **q_1** = il recipiente conteneva una sola biglia, l’ho rimossa e dico “sì”
- **q_2** = il recipiente conteneva due o più biglie, ne ho rimossa una e ora rimuovo l’altra

Automi a stati finiti non deterministici

Definizione

Un **automa a stati finiti non deterministico** (detto anche **NFA**, da Non-deterministic Finite-state Automaton) è una quintupla $A = (Q, \Sigma, \delta, q_0, F)$ dove:

- Q è un insieme finito di **stati**
- Σ è l'**alfabeto** riconosciuto dall'automa
- $\delta : Q \times \Sigma \rightarrow \wp(Q)$ è la **funzione di transizione** (notare il codominio)
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'insieme di **stati finali**

Note

- $\delta(q, a)$ è l'insieme degli stati in cui l'NFA può scegliere di transire quando si trova nello stato q e legge il simbolo a
- se $\delta(q, a)$ è un **singoletto**, c'è una sola scelta (è il caso deterministico)
- se $\delta(q, a)$ è vuoto l'automa **rifiuta** la stringa

Linguaggio riconosciuto da un NFA

Definizione

La **funzione di transizione estesa** dell'NFA $A = (Q, \Sigma, \delta, q_0, F)$ è la funzione $\hat{\delta} : Q \times \Sigma^* \rightarrow \wp(Q)$ definita per induzione sul suo secondo argomento come segue:

$$\hat{\delta}(q, \varepsilon) = \{q\} \quad \hat{\delta}(q, wa) = \{r \in \delta(p, a) \mid p \in \hat{\delta}(q, w)\}$$

Definizione

Il **linguaggio riconosciuto** (o **accettato**) dall'NFA $A = (Q, \Sigma, \delta, q_0, F)$ è denotato da $L(A)$ e definito come segue:

$$L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Nota

- L'NFA riconosce una stringa w se **esiste** un percorso etichettato con w che lo porta dallo stato iniziale q_0 a uno dei suoi stati finali in F .

Rappresentazione tabellare di NFA

Automa in slide 4

Stato	b
$\rightarrow q_0$	$\{q_1, q_2\}$
$*q_1$	\emptyset
q_2	$\{q_0\}$

Osservazioni

- gli insiemi **singololetto** indicano transizioni **deterministiche**
- l'insieme **vuoto** indica che l'NFA “non sa cosa fare” e **rifiuta** la stringa
- gli altri insiemi indicano transizioni **non** deterministiche (scelte)

DFA → NFA

Teorema

Dato un DFA D , esiste un NFA N tale che $L(N) = L(D)$

Dimostrazione

Dato un DFA $D = (Q, \Sigma, \delta_D, q_0, F)$ definiamo $N = (Q, \Sigma, \delta_N, q_0, F)$ dove

$$\delta_N(q, a) = \{\delta_D(q, a)\}$$

Si può dimostrare, per induzione su $|w|$, che

$$\hat{\delta}_D(q_0, w) = p \iff \hat{\delta}_N(q_0, w) = \{p\}$$

da cui si conclude che

$$\hat{\delta}_D(q_0, w) \in F \iff \hat{\delta}_N(q_0, w) \cap F \neq \emptyset$$

Conseguenze

- ogni linguaggio regolare (cioè riconosciuto da un DFA) è riconosciuto da un NFA
- il potere riconoscitivo degli NFA è almeno pari a quello dei DFA

NFA → DFA

Teorema

Dato un NFA N , esiste un DFA D tale che $L(D) = L(N)$

Intuizione

- creiamo un DFA i cui stati sono **insiemi di stati** dell'NFA
- il DFA traccia **tutti gli stati** in cui l'NFA si può trovare durante il riconoscimento di una stringa, ovvero il DFA traccia **tutte le scelte** possibili che l'NFA può fare
- siccome l'NFA ha un numero **finito** di stati (diciamo n), anche gli stati del DFA lo sono (al massimo 2^n)

Conseguenze

- ogni linguaggio riconosciuto da un NFA è **regolare**
- combinando questo risultato e quello della [slide 8](#), concludiamo che NFA e DFA hanno lo **stesso potere riconoscitivo**

NFA → DFA: costruzione per sottoinsiemi

Dato un NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ definiamo $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ dove

- $Q_D = \wp(Q_N)$, ovvero Q_D è l'insieme dei sottoinsiemi di Q_N
- per ogni $S \subseteq Q_N$ e ogni $a \in \Sigma$ definiamo $\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q, a)$
- $F_D = \{S \subseteq Q_N \mid S \cap F_N \neq \emptyset\}$

Se si dimostra l'equazione

$$\hat{\delta}_N(q_0, w) = \hat{\delta}_D(\{q_0\}, w)$$

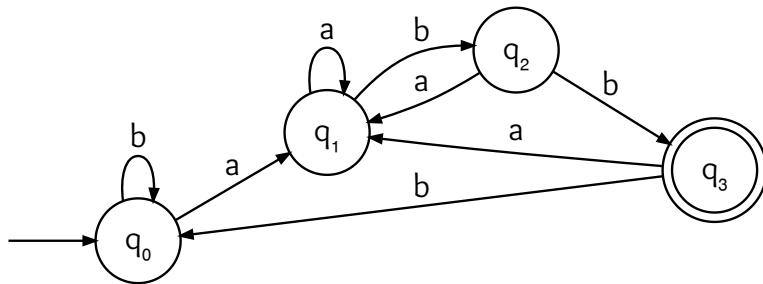
si può concludere che

$$\begin{aligned} w \in L(N) &\iff \hat{\delta}_N(q_0, w) \cap F_N \neq \emptyset && \text{def. di } L(N) \\ &\iff \hat{\delta}_D(\{q_0\}, w) \cap F_N \neq \emptyset && \text{equazione qui sopra} \\ &\iff \hat{\delta}_D(\{q_0\}, w) \in F_D && \text{def. di } F_D \\ &\iff w \in L(D) && \text{def. di } L(D) \end{aligned}$$

La dimostrazione è una semplice induzione su $|w|$ (dettagli nel libro di testo)

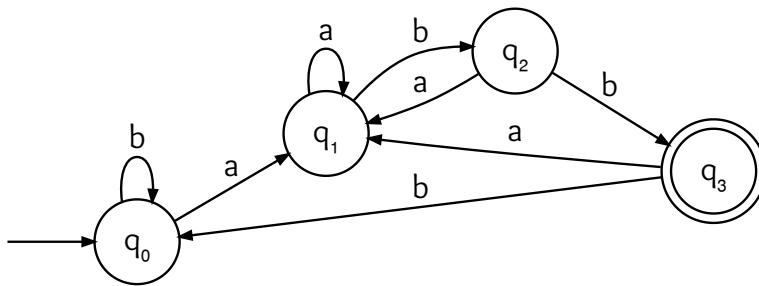
Esempio: stringhe che terminano con abb

Soluzione deterministica

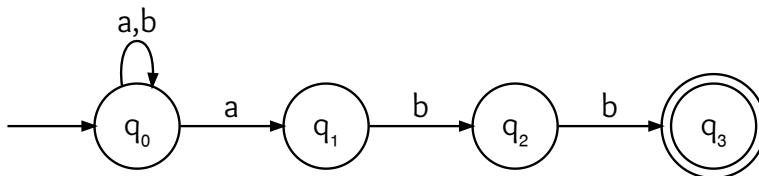


Esempio: stringhe che terminano con abb

Soluzione deterministica



Soluzione non deterministica

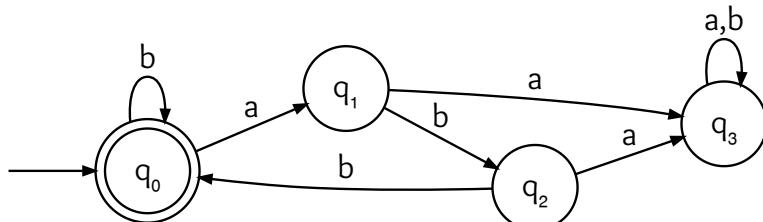


- quando l'automa è nello stato q_0 e legge una a, può **scegliere** se restare in q_0 oppure spostarsi in q_1 e avvicinarsi allo stato finale
- è come se l'automa sapesse qual è la a che annuncia il suffisso abb (quando c'è)
- l'automa non deterministico ha **meno transizioni** di quello deterministico

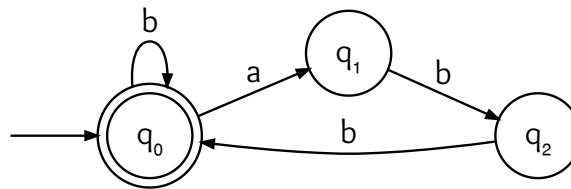
Esempio: ogni a è seguita da bb

Definire un automa che riconosce le stringhe in cui **ogni** a è seguita da bb

Soluzione deterministica

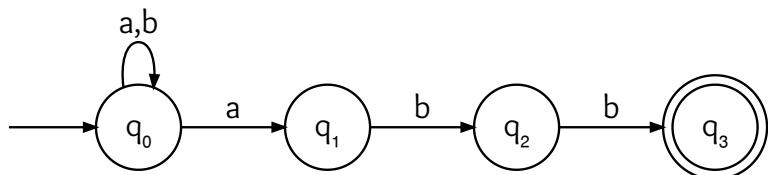


Soluzione non deterministica

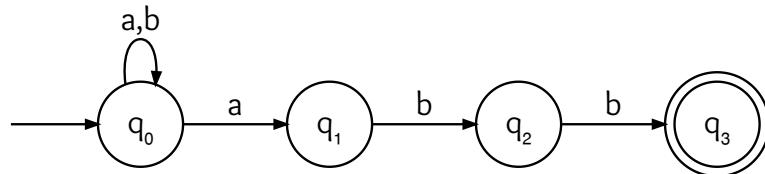


- l'automa non deterministico ha **meno stati e meno transizioni** di quello deterministico

Esempio: stringhe che terminano con abb

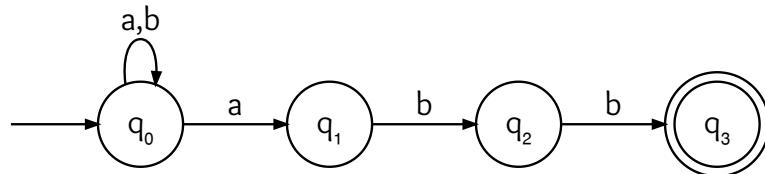


Esempio: stringhe che terminano con abb

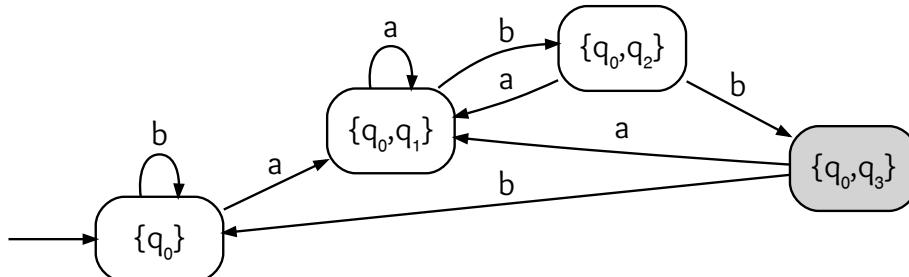


Stato	a	b
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
$*\{q_0, q_3\}$	$\{q_0, q_1\}$	$\{q_0\}$

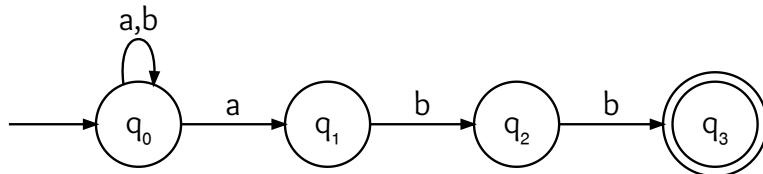
Esempio: stringhe che terminano con abb



Stato	a	b
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
$*\{q_0, q_3\}$	$\{q_0, q_1\}$	$\{q_0\}$



Esempio: stringhe che terminano con abb

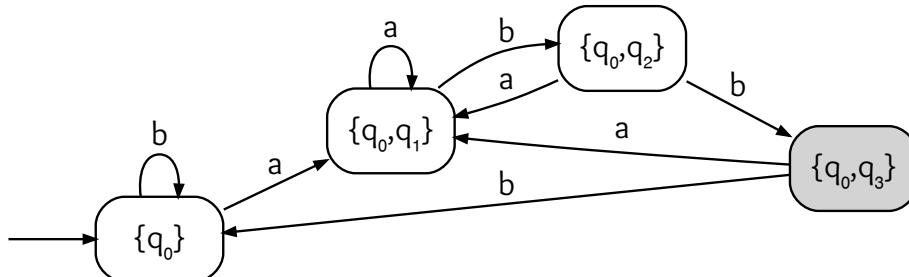


Stato	a	b
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
$*\{q_0, q_3\}$	$\{q_0, q_1\}$	$\{q_0\}$

Anziché considerare **tutti i** sottoinsiemi di stati dell'NFA, scopriamo man mano quelli che sono raggiungibili dallo stato iniziale $\{q_0\}$

Il DFA ottenuto è isomorfo (anche se non identico) a quello della [slide 11](#) (l'unico stato finale ha lo sfondo grigio)

Il **nome** che diamo agli stati **non influenza** il linguaggio riconosciuto



Esercizi

1. Convertire in DFA l'NFA della [slide 12](#)
2. Definire un NFA che riconosce le stringhe di 0 e 1 in cui il terzultimo simbolo è un 1
3. Convertire in DFA l'NFA dell'esercizio precedente
4. Disegnare i diagrammi di transizione dei seguenti NFA e convertirli in DFA

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	\emptyset
$*s$	$\{s\}$	$\{s\}$

	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	\emptyset	$\{p\}$

5. Definire un NFA sull'alfabeto { a, c, e, n, s } che riconosca le parole cane, casa e cena, poi convertirlo in DFA

Linguaggi Formali e Traduttori

2.3 Automi a stati finiti con ϵ -transizioni

- Sommario
- Esempio: costanti numeriche con segno
- Automi a stati finiti con ϵ -transizioni
- ϵ -chiusura
- Esempio di calcolo della ϵ -chiusura
- Linguaggio riconosciuto da un ϵ -NFA
- NFA \rightarrow ϵ -NFA
- ϵ -NFA \rightarrow DFA
- ϵ -NFA \rightarrow DFA: costruzione per sottoinsiemi
- Esempio: costanti numeriche con segno
- Esempio: costruzione modulare di automi
- Esercizi
- Dimostrazioni

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Automi deterministici e non deterministici

Automi in cui ogni **transizione** corrisponde alla **lettura di un simbolo** nella stringa da riconoscere

$$\delta : Q \times \Sigma \rightarrow Q \quad \delta : Q \times \Sigma \rightarrow \wp(Q)$$

Automi con ϵ -transizioni

Automi che possono eseguire **transizioni spontanee** senza leggere alcun simbolo nella stringa da riconoscere

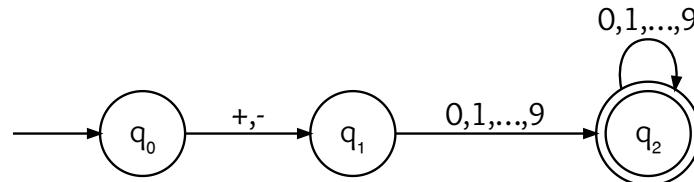
$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$$

In questa lezione

1. Introduciamo la classe degli automi con **ϵ -transizioni**
2. Dimostriamo che ogni linguaggio riconosciuto da un automa con ϵ -transizioni può essere riconosciuto anche da un automa deterministico
3. Usiamo le ϵ -transizioni laddove conveniente, per esempio per rappresentare **parti facoltative** di una stringa da riconoscere o per costruire agevolmente automi complessi in maniera **modulare**

Esempio: costanti numeriche con segno

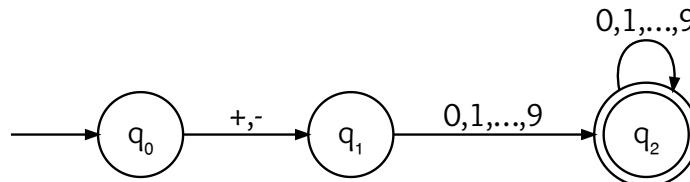
Segno obbligatorio



- quando l'automa è nello stato q_0 si aspetta di riconoscere un segno (+ o -)

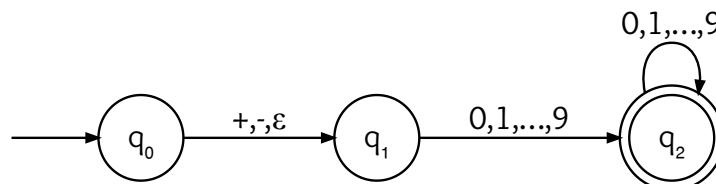
Esempio: costanti numeriche con segno

Segno obbligatorio



- quando l'automa è nello stato q_0 si aspetta di riconoscere un segno (+ o -)

Segno facoltativo



- quando l'automa è nello stato q_0 può riconoscere un segno oppure spostarsi **spontaneamente** in q_1 e avvicinarsi allo stato finale
- abbiamo aggiunto **una transizione** da q_0 a q_1 invece di **10 transizioni** da q_0 a q_2

Automi a stati finiti con ϵ -transizioni

Definizione

Un **automa a stati finiti con ϵ -transizioni** (detto anche **ϵ -NFA**) è una quintupla $A = (Q, \Sigma, \delta, q_0, F)$ dove:

- Q è un insieme finito di **stati**
- Σ è l'**alfabeto** riconosciuto dall'automa
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \wp(Q)$ è la **funzione di transizione** (notare il dominio)
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'insieme di **stati finali**

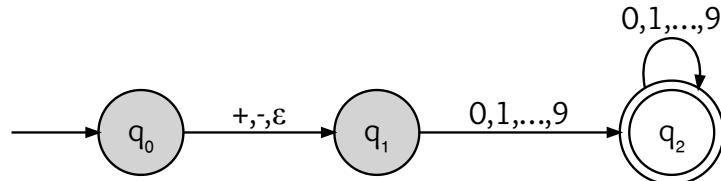
Note

- $\delta(q, a)$ è l'insieme degli stati in cui l' ϵ -NFA può transire quando si trova nello stato q leggendo il simbolo a
- $\delta(q, \epsilon)$ è l'insieme degli stati in cui l' ϵ -NFA può **transire spontaneamente** quando si trova nello stato q , senza leggere alcun simbolo

ϵ -chiusura

Intuizione

Per definire il linguaggio riconosciuto da un ϵ -NFA, è importante riuscire a determinare quali stati sono raggiungibili grazie alle ϵ -transizioni



Definizione

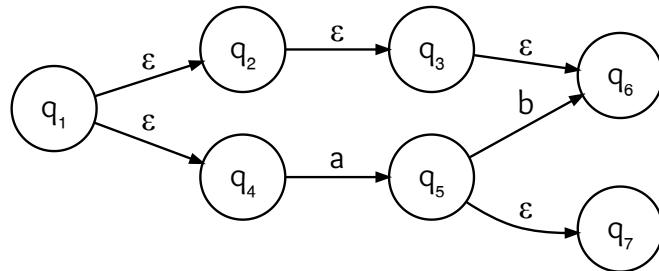
$\text{ECLOSE}(q)$ è il più piccolo insieme di stati tale che:

1. $q \in \text{ECLOSE}(q)$
2. se $p \in \text{ECLOSE}(q)$, allora $\delta(p, \epsilon) \subseteq \text{ECLOSE}(q)$

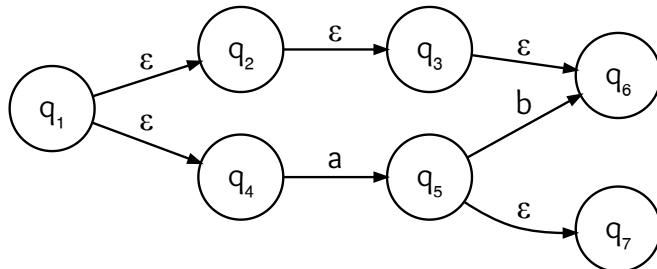
Generalizzazione della ϵ -chiusura a insiemi di stati

Quando S è un insieme di stati, definiamo $\text{ECLOSE}(S) = \bigcup_{q \in S} \text{ECLOSE}(q)$

Esempio di calcolo della ε -chiusura



Esempio di calcolo della ε -chiusura



- $\text{ECLOSE}(q_1) = \{q_1, q_2, q_3, q_4, q_6\}$
- $\text{ECLOSE}(q_2) = \{q_2, q_3, q_6\}$
- $\text{ECLOSE}(q_3) = \{q_3, q_6\}$
- $\text{ECLOSE}(q_4) = \{q_4\}$
- $\text{ECLOSE}(q_5) = \{q_5, q_7\}$
- $\text{ECLOSE}(q_6) = \{q_6\}$
- $\text{ECLOSE}(q_7) = \{q_7\}$

Linguaggio riconosciuto da un ε -NFA

Definizione

La funzione di transizione estesa dell' ε -NFA $A = (Q, \Sigma, \delta, q_0, F)$ è la funzione $\hat{\delta} : Q \times \Sigma^* \rightarrow \wp(Q)$ definita per induzione sul suo secondo argomento come segue:

$$\hat{\delta}(q, \varepsilon) = \text{ECLOSE}(q) \quad \hat{\delta}(q, wa) = \{r \in \text{ECLOSE}(\delta(p, a)) \mid p \in \hat{\delta}(q, w)\}$$

Definizione

Il linguaggio riconosciuto (o accettato) dall' ε -NFA $A = (Q, \Sigma, \delta, q_0, F)$ è denotato da $L(A)$ e definito come segue:

$$L(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Nota

- L' ε -NFA riconosce una stringa w se esiste un percorso etichettato con w che lo porta dallo stato iniziale q_0 a uno dei suoi stati finali in F

NFA → ϵ -NFA

Teorema

Dato un NFA N , esiste un ϵ -NFA E tale che $L(E) = L(N)$

Dimostrazione

Basta osservare che un NFA è un caso particolare di ϵ -NFA in cui non ci sono ϵ -transizioni

Conseguenze

- ogni linguaggio regolare (cioè riconosciuto da un DFA) è riconosciuto da un ϵ -NFA
- il potere riconoscitivo degli ϵ -NFA è almeno pari a quello dei DFA/NFA, che abbiamo già dimostrato essere equivalenti

ϵ -NFA → DFA

Teorema

Dato un ϵ -NFA E , esiste un DFA D tale che $L(D) = L(E)$

Intuizione

- usiamo la costruzione per sottoinsiemi come nel caso NFA → DFA
- occorre fare attenzione agli stati raggiungibili da ϵ -transizioni
- possiamo usare la nozione di ϵ -chiusura!

Conseguenze

- ogni linguaggio riconosciuto da un ϵ -NFA è **regolare**
- combinando questo risultato e quello della [slide 8](#), concludiamo che ϵ -NFA, NFA e DFA hanno lo stesso potere riconoscitivo

ε -NFA \rightarrow DFA: costruzione per sottoinsiemi

Dato un ε -NFA $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ definiamo $D = (Q_D, \Sigma, \delta_D, \text{ECLOSE}(q_0), F_D)$ dove

- $Q_D = \wp(Q_E)$, ovvero Q_D è l'insieme dei sottoinsiemi di Q_E
- per ogni $S \subseteq Q_E$ e ogni $a \in \Sigma$ definiamo $\delta_D(S, a) = \bigcup_{q \in S} \text{ECLOSE}(\delta_E(q, a))$
- $F_D = \{S \subseteq Q_E \mid S \cap F_E \neq \emptyset\}$

Se si dimostra l'equazione

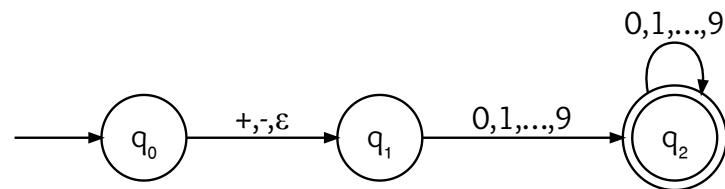
$$\hat{\delta}_E(q_0, w) = \hat{\delta}_D(\text{ECLOSE}(q_0), w)$$

si può concludere che

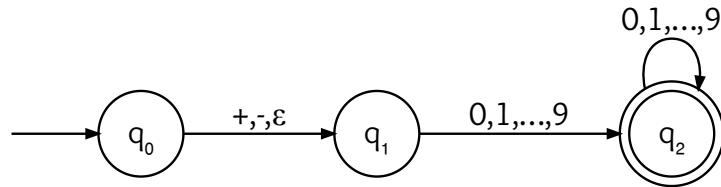
$$\begin{aligned} w \in L(E) &\iff \hat{\delta}_E(q_0, w) \cap F_E \neq \emptyset && \text{def. di } L(E) \\ &\iff \hat{\delta}_D(\text{ECLOSE}(q_0), w) \cap F_E \neq \emptyset && \text{equazione qui sopra} \\ &\iff \hat{\delta}_D(\text{ECLOSE}(q_0), w) \in F_D && \text{def. di } F_D \\ &\iff w \in L(D) && \text{def. di } L(D) \end{aligned}$$

Dettagli nel libro di testo.

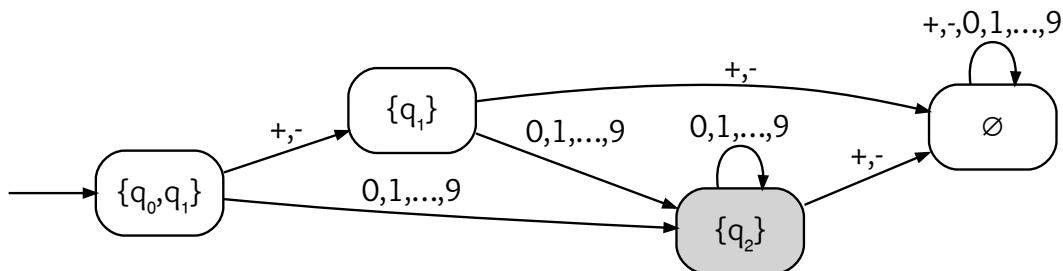
Esempio: costanti numeriche con segno



Esempio: costanti numeriche con segno



	$+,-$	$0,1,...,9$
$\rightarrow \{q_0, q_1\}$	$\{q_1\}$	$\{q_2\}$
$\{q_1\}$	\emptyset	$\{q_2\}$
$*\{q_2\}$	\emptyset	$\{q_2\}$
\emptyset	\emptyset	\emptyset



Esempio: costruzione modulare di automi

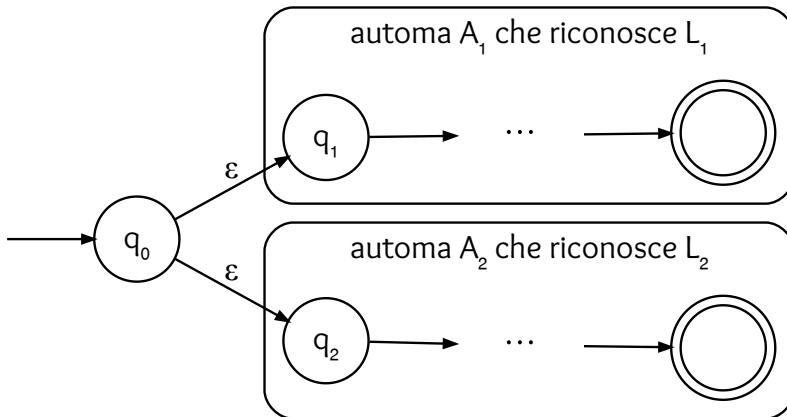
Teorema

I linguaggi regolari sono **chiusi** rispetto all'operazione di **unione**, ovvero se L_1 ed L_2 sono regolari, allora anche $L_1 \cup L_2$ è regolare

Dimostrazione

Se L_1 ed L_2 sono regolari, allora esistono due automi a stati finiti A_1 e A_2 tali che $L_1 = L(A_1)$ e $L_2 = L(A_2)$. Supponiamo $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ e $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Costruiamo $A = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup F_2)$ dove



$$\delta(q, \alpha) = \begin{cases} \{q_1, q_2\} & \text{se } q = q_0 \text{ e } \alpha = \varepsilon \\ \delta_1(q, \alpha) & \text{se } q \in Q_1 \\ \delta_2(q, \alpha) & \text{se } q \in Q_2 \\ \emptyset & \text{altrimenti} \end{cases}$$

Abbiamo che $L(A) = L(A_1) \cup L(A_2)$

Esercizi

1. Definire un ϵ -NFA che riconosca le stringhe composte da 0 o più a, seguite da 0 o più b, seguite da 0 o più c.
2. Definire un ϵ -NFA che riconosca le stringhe formate da 01 ripetuto una o più volte, o da 010 ripetuto una o più volte.
3. Per l' ϵ -NFA rappresentato in forma tabellare qui sotto, calcolare l' ϵ -chiusura di ciascuno stato, descrivere sommariamente il linguaggio accettato e convertire l'automa in DFA.

	ϵ	a	b	c
$\rightarrow p$	\emptyset	$\{p\}$	$\{q\}$	$\{r\}$
q	$\{p\}$	$\{q\}$	$\{r\}$	\emptyset
$*r$	$\{q\}$	$\{r\}$	\emptyset	$\{p\}$

4. Per l' ϵ -NFA rappresentato in forma tabellare qui sotto, calcolare l' ϵ -chiusura di ciascuno stato, descrivere sommariamente il linguaggio accettato e convertire l'automa in DFA.

	ϵ	a	b	c
$\rightarrow p$	$\{q, r\}$	\emptyset	$\{q\}$	$\{r\}$
q	\emptyset	$\{p\}$	$\{r\}$	$\{p, q\}$
$*r$	\emptyset	\emptyset	\emptyset	\emptyset

Dimostrazioni

1. Dimostrare che per ogni DFA esiste un ϵ -NFA equivalente (cioè che riconosce lo stesso linguaggio) che ha esattamente **uno** stato finale
2. Dimostrare che i linguaggi regolari sono **chiusi** rispetto all'operazione di **concatenazione**.
Suggerimento: se utile usare il risultato dimostrato nell'esercizio precedente

Linguaggi Formali e Traduttori

2.4 Espressioni regolari

- Sommario
- Sintassi delle espressioni regolari
- Significato di un'espressione regolare
- Proprietà delle espressioni regolari
- Espressioni e linguaggi regolari
- Espressione regolare → ϵ -NFA (1/4)
- Espressione regolare → ϵ -NFA (2/4)
- Espressione regolare → ϵ -NFA (3/4)
- Espressione regolare → ϵ -NFA (4/4)
- Esempio: sequenze di a seguite da sequenze di b
- Esempio: 0 oppure sequenze non vuote di 1
- Esempio: ogni a è seguita da bb
- Esempio: esiste a seguita da bb
- Esercizi sulla definizione di espressioni regolari
- Esercizi sulla conversione di espressione regolari

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Automi

- approcci riconoscitivi per descrivere linguaggi regolari
- 3 varianti equivalenti: deterministici, non deterministici, con ϵ -transizioni

Espressioni regolari

- approccio generativo per descrivere linguaggi regolari

In questa lezione

1. Definiamo la sintassi ed il significato delle espressioni regolari
2. Enunciamo alcune leggi fondamentali delle espressioni regolari
3. Mostriamo che le espressioni regolari generano tutti e soli i linguaggi regolari

Sintassi delle espressioni regolari

Definizione

Le **espressioni regolari** su un alfabeto Σ (abbreviate **RE**, da Regular Expressions) sono definite induttivamente come segue:

- \emptyset ed ϵ sono espressioni regolari;
- se $a \in \Sigma$, allora a è un'espressione regolare;
- se E ed F sono espressioni regolari, allora $E + F$ ed EF sono espressioni regolari;
- se E è un'espressione regolare, allora E^* è un'espressione regolare.

Convenzioni

- assumiamo la **precedenza** degli operatori $+$ < concatenazione $< ^*$
- usiamo le **parentesi** per disambiguare la struttura di un'espressione regolare

Esempi

- $ab + c = (ab) + c \neq a(b + c)$
- $01^* = 0(1^*) \neq (01)^*$
- $0 + 11^* = 0 + (1(1^*))$

Significato di un'espressione regolare

Se E è un'espressione regolare, il **linguaggio generato** da E , denotato da $L(E)$, è definito per induzione sulla struttura di E come segue:

$L(\emptyset)$	$=$	\emptyset	linguaggio vuoto
$L(\varepsilon)$	$=$	$\{\varepsilon\}$	stringa vuota
$L(a)$	$=$	$\{a\}$	simbolo dell'alfabeto
$L(E + F)$	$=$	$L(E) \cup L(F)$	unione
$L(EF)$	$=$	$L(E)L(F)$	concatenazione
$L(E^*)$	$=$	$L(E)^*$	chiusura di Kleene

Diciamo che E ed F sono **equivalenti**, notazione $E = F$, se $L(E) = L(F)$.

Esercizio

Calcolare il linguaggio generato dalle espressioni regolari $(a + b)^*$ e $(ab)^*$.

Proprietà delle espressioni regolari

Unione

- commutatività: $E + F = F + E$
- associatività: $E + (F + G) = (E + F) + G$
- idempotenza: $E + E = E$
- identità: $E + \emptyset = \emptyset + E = E$

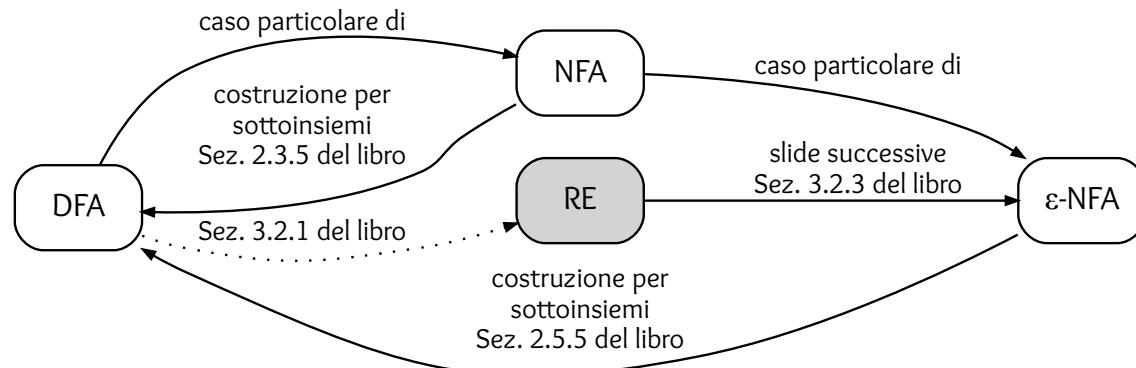
Concatenazione

- associatività: $E(FG) = (EF)G$
- identità: $E\epsilon = \epsilon E = E$
- assorbimento: $E\emptyset = \emptyset E = \emptyset$
- distributività sinistra della concatenazione sull'unione: $E(F + G) = EF + EG$
- distributività destra della concatenazione sull'unione: $(E + F)G = EG + FG$

Chiusura di Kleene

- idempotenza: $(E^*)^* = E^*$
- casi banali: $\epsilon^* = \emptyset^* = \epsilon$

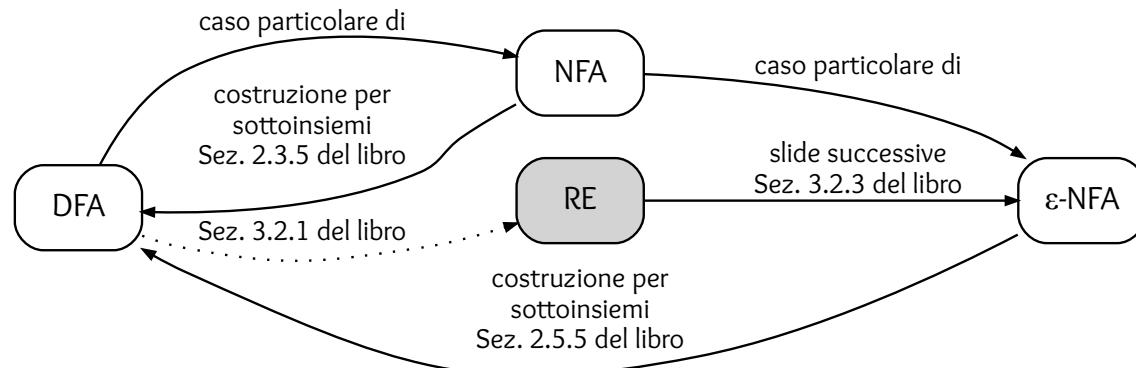
Espressioni e linguaggi regolari



Conseguenza

DFA, NFA, ϵ -NFA ed espressioni regolari sono approcci diversi ma **equivalenti** di definire (riconoscere, generare) linguaggi regolari

Espressioni e linguaggi regolari



Conseguenza

DFA, NFA, ϵ -NFA ed espressioni regolari sono approcci diversi ma **equivalenti** di definire (riconoscere, generare) linguaggi regolari

Teorema

Per ogni DFA A , esiste un'espressione regolare E tale che $L(A) = L(E)$.

Dimostrazione

Si veda la Sez. 3.2.1 del libro (lettura facoltativa)

Espressione regolare $\rightarrow \varepsilon$ -NFA (1/4)

Teorema

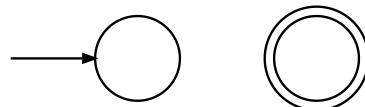
Data un'espressione regolare E , esiste un ε -NFA A tale che $L(A) = L(E)$.

Dimostrazione

Costruiamo A per induzione sulla struttura di E e per casi sulla sua forma, facendo in modo che l' ε -NFA ottenuto abbia sempre **esattamente uno stato finale** (quello più a destra nei diagrammi che seguono).

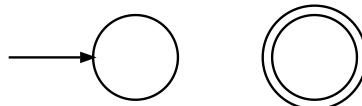
Espressione regolare → ϵ -NFA (2/4)

Caso \emptyset (linguaggio vuoto)

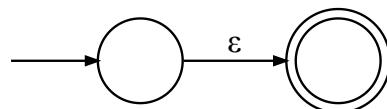


Espressione regolare $\rightarrow \epsilon$ -NFA (2/4)

Caso \emptyset (linguaggio vuoto)

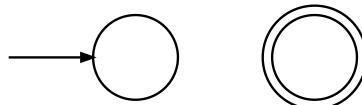


Caso ϵ (linguaggio che contiene la sola stringa vuota)

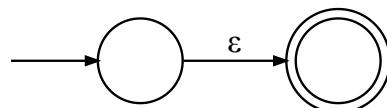


Espressione regolare $\rightarrow \epsilon$ -NFA (2/4)

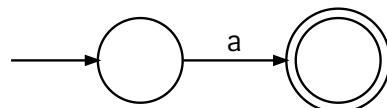
Caso \emptyset (linguaggio vuoto)



Caso ϵ (linguaggio che contiene la sola stringa vuota)

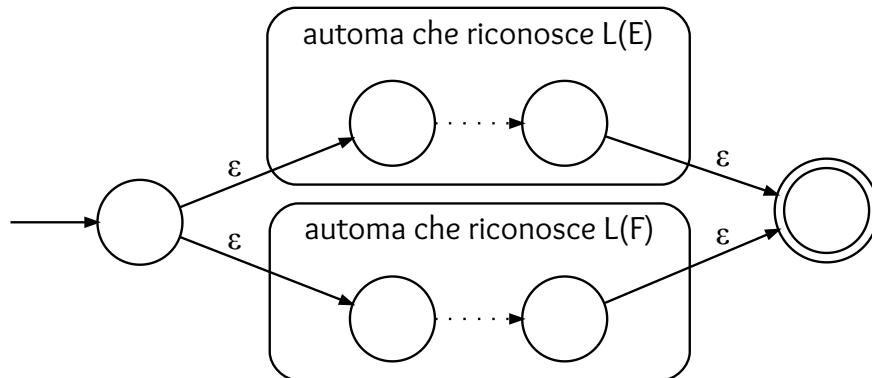


Caso a (linguaggio che contiene solo a)



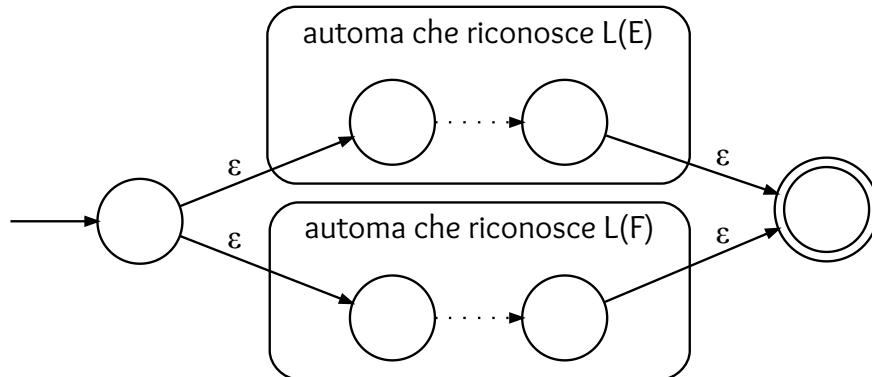
Espessione regolare $\rightarrow \epsilon$ -NFA (3/4)

Caso $E + F$ (unione)

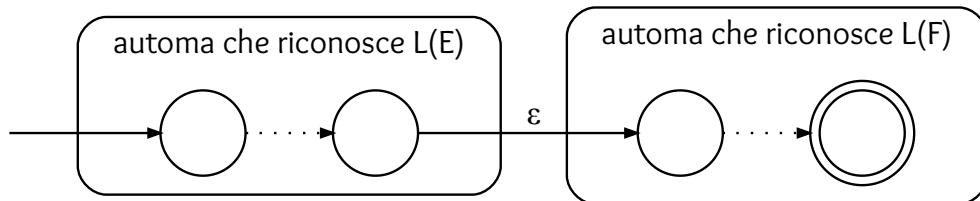


Espressione regolare $\rightarrow \varepsilon$ -NFA (3/4)

Caso $E + F$ (unione)

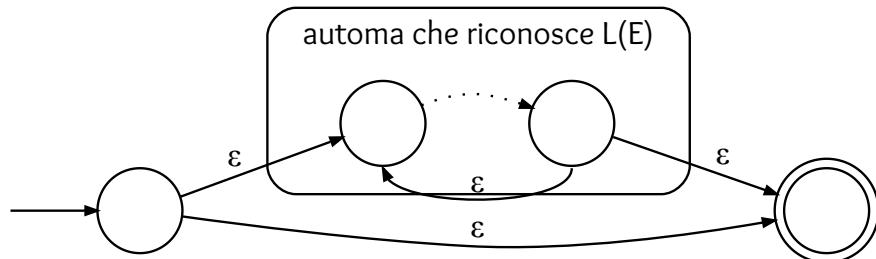


Caso EF (concatenazione)



Espessione regolare $\rightarrow \epsilon$ -NFA (4/4)

Caso E^* (chiusura di Kleene)



Esempio: sequenze di a seguite da sequenze di b

$$\begin{aligned}L(a^*b^*) &= L(a^*)L(b^*) \\&= L(a)^*L(b)^* \\&= \{a\}^*\{b\}^* \\&= \{\varepsilon, a, aa, aaa, \dots\}\{\varepsilon, b, bb, bbb, \dots\} \\&= \{\varepsilon, a, b, aa, ab, bb, aaa, aab, abb, bbb, \dots\}\end{aligned}$$

Esempio: 0 oppure sequenze non vuote di 1

$$\begin{aligned}L(0 + 11^*) &= L(0) \cup L(11^*) \\&= \{0\} \cup L(1)L(1^*) \\&= \{0\} \cup \{1\}L(1)^* \\&= \{0\} \cup \{1\}\{1\}^* \\&= \{0\} \cup \{1\}\{\varepsilon, 1, 11, 111, \dots\} \\&= \{0\} \cup \{1, 11, 111, 1111, \dots\}\end{aligned}$$

Esempio: ogni a è seguita da bb

$$\begin{aligned}L((abb + b)^*) &= L(abb + b)^* \\&= (L(abb) \cup L(b))^* \\&= (L(a)L(b)L(b) \cup L(b))^* \\&= (\{a\}\{b\}\{b\} \cup \{b\})^* \\&= (\{abb\} \cup \{b\})^* \\&= \{abb, b\}^*\end{aligned}$$

Esempio: esiste a seguita da bb

$$\begin{aligned}L((a + b)^*abb(a + b)^*) &= L((a + b)^*)L(abb)L((a + b)^*) \\&= L(a + b)^*L(a)L(b)L(b)L(a + b)^* \\&= (L(a) \cup L(b))^*L(a)L(b)(L(a) \cup L(b))^* \\&= (\{a\} \cup \{b\})^*\{a\}\{b\}\{b\}(\{a\} \cup \{b\})^* \\&= \{a, b\}^*\{abb\}\{a, b\}^* \\&= \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}\{abb\}\{\varepsilon, a, b, aa, ab, ba, bb, \dots\}\end{aligned}$$

Esercizi sulla definizione di espressioni regolari

Definire espressioni regolari che generino i seguenti linguaggi:

1. stringhe di a, b e c che iniziano con due a e finiscono con due b
2. stringhe di 0 e 1 la cui lunghezza è un multiplo di 3
3. stringhe di 0 e 1 con un numero pari di 0
4. stringhe di a, b e c che **non contengono** la sottostringa ab
5. costanti numeriche binarie pari senza 0 inutili a sinistra (es. 0, 10, ma non 010 o 11)
6. costanti numeriche decimali con virgola facoltativa (es. 42, .5, 12.3, 12. ma non .)

Esercizi sulla conversione di espressione regolari

Convertire le seguenti espressioni regolari in ϵ -NFA e gli automati ottenuti in DFA:

1. $(a + b)^*$
2. $(ab)^*$
3. a^*b^*
4. $a^* + b^*$

Linguaggi Formali e Traduttori

2.5 Proprietà di chiusura dei linguaggi regolari

- Sommario
- Unione e concatenazione
- Complemento
- Intersezione
- Intersezione (costruzione diretta)
- Differenza
- Inversione
- Esercizi

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

In questa lezione studiamo le più importanti proprietà di chiusura dei linguaggi regolari ponendoci la seguente domanda: dati due linguaggi regolari L ed L' , i seguenti linguaggi sono regolari?

- $L \cup L'$
- $L \cap L'$
- LL'
- \overline{L}
- $L - L'$
- L^R

In tutti i casi possiamo rispondere affermativamente.

Unione e concatenazione

Teorema

I linguaggi regolari sono chiusi per unione e concatenazione.

Dimostrazione

Siano L_1 ed L_2 linguaggi regolari.

Dunque esistono due espressioni regolari E_1 ed E_2 tali che $L_1 = L(E_1)$ e $L_2 = L(E_2)$.

Ora $E_1 + E_2$ e E_1E_2 sono espressioni regolari che generano, rispettivamente, $L_1 \cup L_2$ e L_1L_2 .

Concludiamo che $L_1 \cup L_2$ e L_1L_2 sono regolari.

Complemento

Teorema

I linguaggi regolari sono chiusi per complemento.

Dimostrazione

Sia L un linguaggio regolare.

Dunque esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ tale che $L = L(A)$.

Definiamo $B = (Q, \Sigma, \delta, q_0, Q - F)$ e osserviamo che

$$w \in L(A) \Leftrightarrow \hat{\delta}(q_0, w) \in F \Leftrightarrow w \notin L(B)$$

Concludiamo che $\overline{L} = L(B)$ e che \overline{L} è regolare.

Intersezione

Teorema

I linguaggi regolari sono chiusi per intersezione.

Dimostrazione

Siano L_1 ed L_2 linguaggi regolari su un alfabeto Σ .

Usando le **leggi di De Morgan**, osserviamo che

$$L_1 \cap L_2 = \overline{\overline{L_1} \cap \overline{L_2}} = \overline{\overline{L_1} \cup \overline{L_2}}$$

Siccome i linguaggi regolari sono chiusi per unione e complemento, concludiamo che $L_1 \cap L_2$ è regolare.

Intersezione (costruzione diretta)

Dimostrazione alternativa

Siano L_1 ed L_2 linguaggi regolari su un alfabeto Σ .

Dunque esistono due DFA $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ e $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ tali che $L_1 = L(A_1)$ e $L_2 = L(A_2)$.

Definiamo $B = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$ dove

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$$

per ogni $p \in Q_1$, $q \in Q_2$ e $a \in \Sigma$. Concludiamo osservando che

$$\begin{aligned} w \in L(B) &\Leftrightarrow \hat{\delta}((q_1, q_2), w) \in F_1 \times F_2 \\ &\Leftrightarrow \hat{\delta}_1(q_1, w) \in F_1 \wedge \hat{\delta}_2(q_2, w) \in F_2 \\ &\Leftrightarrow w \in L(A_1) \wedge w \in L(A_2) \\ &\Leftrightarrow w \in L_1 \cap L_2 \end{aligned}$$

Differenza

Teorema

I linguaggi regolari sono chiusi per differenza.

Dimostrazione

Siano L_1 ed L_2 linguaggi regolari su un alfabeto Σ .

Per concludere basta osservare che $L_1 - L_2 = L_1 \cap \overline{L_2}$ e ricordare che i linguaggi regolari sono chiusi per intersezione e complemento.

Inversione

Teorema

I linguaggi regolari sono chiusi per inversione.

Dimostrazione

Se L è un linguaggio regolare deve esistere un'espressione regolare E tale che $L = L(E)$.

Definiamo l'espressione regolare E^R per induzione sulla struttura di E e per casi sulla sua forma, usando le seguenti equazioni:

$$\emptyset^R = \emptyset$$

$$\varepsilon^R = \varepsilon$$

$$a^R = a$$

$$(E_1 + E_2)^R = E_1^R + E_2^R$$

$$(E_1 E_2)^R = E_2^R E_1^R$$

$$(E^*)^R = (E^R)^*$$

È facile dimostrare che $L(E^R) = L(E)^R$, dunque L^R è regolare.

Esercizi

Autandosi con le tecniche illustrate in questa lezione, risolvere i seguenti esercizi.

1. Se L è un linguaggio regolare, cosa si può dire di L^* e di L^+ ? Sono regolari?
2. Definire un DFA che riconosca il linguaggio delle stringhe sull'alfabeto $\{0, 1\}$ in cui non c'è uno 0 seguito da due 1.
3. Definire un DFA che riconosca il linguaggio delle stringhe sull'alfabeto $\{0, 1\}$ di lunghezza pari e che contengono almeno un 1.
4. Definire un'espressione regolare per il linguaggio $L(E)^R$, dove $E = (ab)^*(b + a^*)^*$.

Linguaggi Formali e Traduttori

2.6 Pumping lemma per i linguaggi regolari

- Sommario
- Linguaggi **non** regolari
- Pumping lemma per linguaggi regolari
- Esempio: $a^k b^k$ non è regolare
- Pumping lemma: dimostrazione (1/3)
- Pumping lemma: dimostrazione (2/3)
- Pumping lemma: dimostrazione (3/3)
- Esempio: $a^k b^m$ con $k \leq m$ non è regolare
- Esempio: a^k con k primo non è regolare
- Esercizi e quesiti

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Per dimostrare che un linguaggio **è regolare**, basta esibire un automa a stati finiti (DFA, NFA o ϵ -NFA) che lo riconosce, oppure una espressione regolare che lo genera. L'incapacità di trovare siffatto automa o siffatta espressione non è una dimostrazione del fatto che il linguaggio non è regolare.

In questa lezione rispondiamo alle seguenti domande:

1. Esistono linguaggi **non** regolari?
2. Se sì, come dimostro che un linguaggio **non** è regolare?

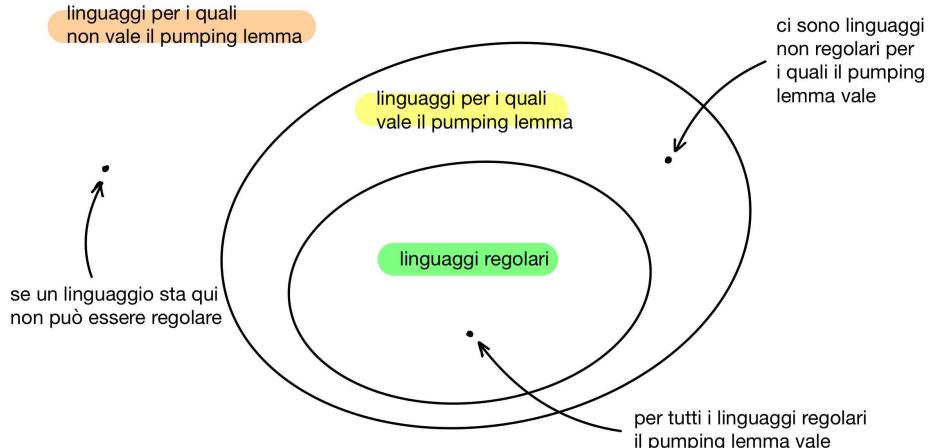
Linguaggi non regolari

- Cerchiamo una proprietà P soddisfatta da tutti i linguaggi regolari:

$$L \text{ regolare} \Rightarrow L \text{ soddisfa } P$$

- Se troviamo un linguaggio L che **non soddisfa P** , allora per **contrapposizione** possiamo concludere che L **non è regolare**:

$$L \text{ non soddisfa } P \Rightarrow L \text{ non è regolare}$$



Pumping lemma per linguaggi regolari

Teorema

Per ogni linguaggio regolare L esiste $n \in \mathbb{N}$ tale che, per ogni $w \in L$ con $|w| \geq n$, esistono x, y e z tali che $w = xyz$ e inoltre:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. $xy^kz \in L$ per ogni $k \geq 0$.

In prosa

- Ogni stringa w “sufficientemente lunga” ($|w| \geq n$) di un linguaggio regolare L ...
- ... contiene una sottostringa non vuota ($y \neq \epsilon$) ...
- ... e “non troppo distante” dall’inizio di w ($|xy| \leq n$) ...
- ... che può essere eliminata ($k = 0$) o replicata a piacere ($k > 0$) ...
- ... consentendoci di trovare altre stringhe di L ($xy^kz \in L$)

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 2 sappiamo che x e y sono composte di sole a.

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 2 sappiamo che x e y sono composte di sole a.

Dalla condizione 1 sappiamo che y contiene almeno una a.

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 2 sappiamo che x e y sono composte di sole a.

Dalla condizione 1 sappiamo che y contiene almeno una a.

Dalla condizione 3 sappiamo che $xz \in L$.

Esempio: $a^k b^k$ non è regolare

Dimostriamo che $L = \{a^k b^k \mid k \geq 0\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 2 sappiamo che x e y sono composte di sole a.

Dalla condizione 1 sappiamo che y contiene almeno una a.

Dalla condizione 3 sappiamo che $xz \in L$.

Ma ora in xz ci sono più b che a, il che contraddice la definizione di L .

Pumping lemma: dimostrazione (1/3)

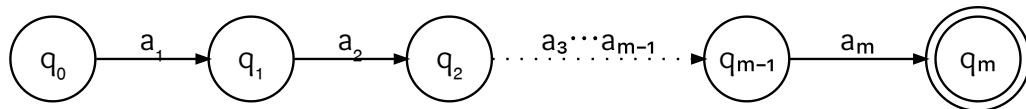
Sia L un linguaggio regolare.

Dunque esiste un DFA $A = (Q, \Sigma, \delta, q_0, F)$ tale che $L = L(A)$.

Poniamo $n = |Q|$, ovvero n è il numero di stati di un DFA che riconosce L .

Prendiamo $w \in L$ tale che $|w| \geq n$. Deve essere $w = a_1a_2 \cdots a_m$ con $m \geq n$.

Se rappresentiamo il cammino fatto da A per riconoscere w come segue

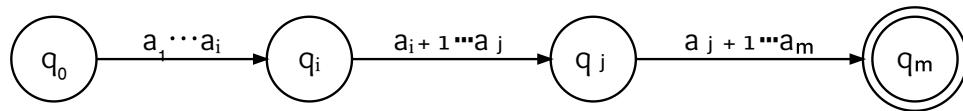


notiamo che questo cammino passa attraverso $m + 1$ stati.

Siccome $m \geq n$ abbiamo $m + 1 > n$. Ovvero, gli stati traversati non possono essere tutti distinti, perché l'automa ne ha solo n .

Pumping lemma: dimostrazione (2/3)

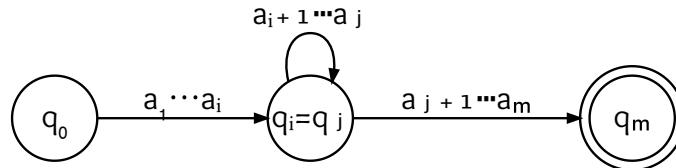
Deduciamo che il cammino fatto da A può essere rappresentato così



dove $q_i = q_j$ e $i < j$.

Possiamo supporre, senza perdere in generalità, che $q_i = q_j$ sia il primo stato che si ripete in questo cammino (ce ne possono essere tanti).

Il cammino fatto da A può allora essere rappresentato anche così:



Pumping lemma: dimostrazione (3/3)

Ora definiamo le stringhe x , y e z come segue:

- $x = a_1a_2 \cdots a_i$
- $y = a_{i+1}a_{i+2} \cdots a_j$
- $z = a_{j+1}a_{j+2} \cdots a_m$

Notiamo che

1. $y \neq \varepsilon$, in quanto $i < j$ dunque in y c'è almeno un simbolo (l'automa è deterministico e non ha ε -transizioni)
2. $|xy| \leq n$ in quanto $q_i = q_j$ è il primo stato che si ripete e quindi gli stati da q_0 a q_j sono al massimo $n + 1$, attraversati leggendo al massimo n simboli di w
3. $xy^kz \in L$ per ogni $k \geq 0$ in quanto tutti i cammini etichettati con xy^kz portano l'automa da q_0 (lo stato iniziale) a q_m (uno stato finale)

e questo conclude la dimostrazione.

Esempio: $a^k b^m$ con $k \leq m$ non è regolare

Dimostriamo che $L = \{a^k b^m \mid 0 \leq k \leq m\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Considero la stringa $w = a^n b^n$, che è in L e ha la proprietà $|w| = 2n \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Dalla condizione 2 sappiamo che x e y sono composte di sole a.

Dalla condizione 1 sappiamo che y contiene almeno una a.

Dalla condizione 3 sappiamo che $xyyz \in L$.

Ma ora in $xyyz$ ci sono più a che b, il che contraddice la definizione di L .

Esempio: a^k con k primo non è regolare

Dimostriamo che $L = \{a^k \mid k \text{ primo}\}$ non è regolare facendo vedere che per L il pumping lemma non vale.

Supponiamo, per assurdo, che esista n con le proprietà enunciate nella [slide 4](#).

Consideriamo la stringa $w = a^p$ dove p è un numero primo $p \geq n + 2$. Siamo sempre in grado di trovare p con questa proprietà in quanto esistono infiniti numeri primi. Inoltre, la stringa w è in L e ha la proprietà $|w| \geq n$.

Devono esistere x , y e z tali che $w = xyz$ e che soddisfano le condizioni 1–3 della [slide 4](#).

Definiamo $m = |y|$, da cui segue che $|xz| = p - m$. Dalle condizioni 1 e 2 sappiamo che $1 \leq m \leq n$. Dalla condizione 3 sappiamo che $xy^{p-m}z \in L$. Tuttavia

$$|xy^{p-m}z| = |xz| + (p - m)|y| = p - m + (p - m)m = (p - m)(m + 1)$$

e ora concludiamo che $|xy^{p-m}z|$ non è primo in quanto:

- da $1 \leq m$ deduco $2 \leq m + 1$, e
- da $m \leq n$ e $p \geq n + 2$ deduco $p - m \geq 2$.

Esercizi e quesiti

Linguaggi non regolari

Dimostrare che i seguenti linguaggi non sono regolari:

1. $\{0^{k^2} \mid k \geq 1\}$
2. $\{0^k 1 0^k \mid k \geq 1\}$
3. $\{0^k 1^{2k} \mid k \geq 1\}$
4. $\{ww \mid w \in \{0,1\}^*\}$
5. $\{ww^R \mid w \in \{0,1\}^*\}$
6. $\{0^k 1^m \mid k \neq m\}$ (suggerimento: usare una proprietà di chiusura)

Proprietà di linguaggi

1. Se \mathcal{L}_i con $i \in \mathbb{N}$ è una famiglia **infinita** di linguaggi regolari, cosa si può dire di $\bigcup_{i \in \mathbb{N}} \mathcal{L}_i$? È sempre un linguaggio regolare?
2. Se \mathcal{L} è un linguaggio **finito**, mostrare come scegliere la costante n del pumping lemma senza sapere come è fatto l'automa che riconosce \mathcal{L} .

Linguaggi Formali e Traduttori

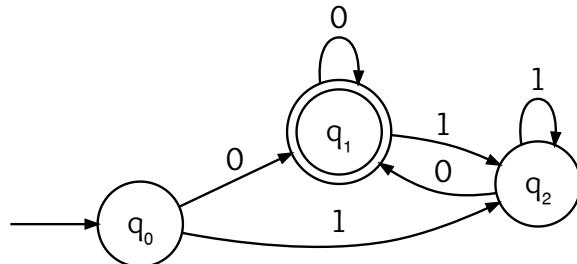
2.7 Minimizzazione di automi a stati finiti deterministici

- Sommario
- Stati (in)distinguibili
- Esempio
- Esempio (dal libro)
- Algoritmo per trovare stati distinguibili
- Esempio
- L'indistinguibilità come equivalenza
- Costruzione dell'automa minimo
- Esempio
- Equivalenza di automi
- Esercizi di minimizzazione
- Esercizi di equivalenza

È proibito condividere e divulgare in qualsiasi forma i materiali didattici caricati sulla piattaforma e le lezioni svolte in videoconferenza: ogni azione che viola questa norma sarà denunciata agli organi di Ateneo e perseguita a termini di legge.

Sommario

Motivazione



- Osservando attentamente questo DFA ci accorgiamo c'è una **ridondanza**
- Una stringa accettata partendo da q_0 è accettata anche partendo da q_2 e viceversa
- Possiamo "fondere" q_0 e q_2 senza alterare il linguaggio riconosciuto dall'automa

Problema

- In generale, è desiderabile lavorare con DFA "piccoli"
- Avendo a disposizione un DFA, è possibile individuarne gli stati ridondanti e "minimizzarlo"?

In questa lezione

- Definiamo un algoritmo che costruisce il DFA **minimo** (cioè con il più piccolo numero di stati) che riconosce un certo linguaggio regolare.

Stati (in)distinguibili

Definizione

Dato un DFA $A = (Q, \Sigma, \delta, q_0, F)$, diciamo che $p, q \in Q$ sono **indistinguibili** se

$$\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$$

per ogni $w \in \Sigma^*$.

Nota

Affinché p e q siano indistinguibili non è necessario che $\hat{\delta}(p, w)$ e $\hat{\delta}(q, w)$ siano lo stesso stato per ogni w , ma solo che siano entrambi finali o entrambi non finali per ogni w .

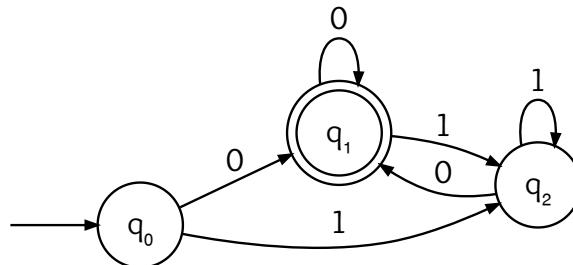
Definizione

Diciamo che due stati $p, q \in Q$ sono **distinguibili** se esiste $w \in \Sigma^*$ tale che solo uno tra $\hat{\delta}(p, w)$ e $\hat{\delta}(q, w)$ appartiene a F . Diciamo che una stringa w con questa proprietà **distingue** p da q .

Nota

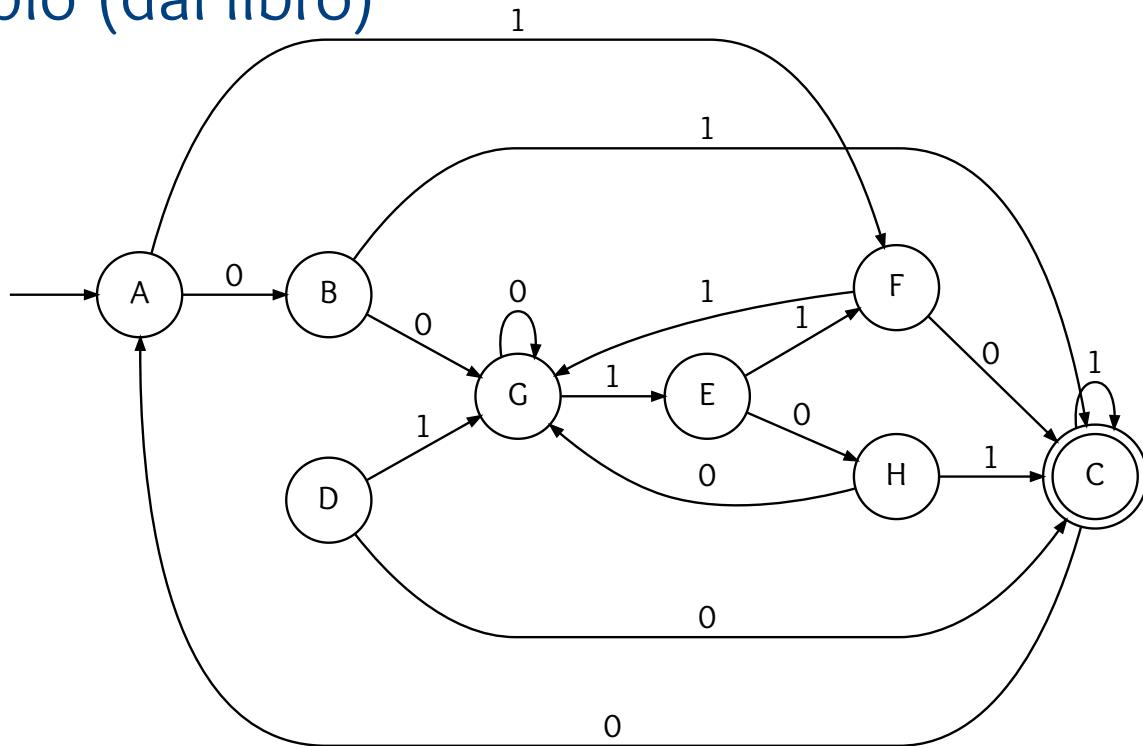
In generale ci possono essere molte stringhe che distinguono due stati.

Esempio



- La stringa ϵ distingue q_0 e q_1 , in quanto q_1 è finale mentre q_0 no.
- Per lo stesso motivo, la stringa ϵ distingue q_1 e q_2 .
- q_0 e q_2 sono indistinguibili, infatti:
 - sono entrambi non finali, dunque ϵ non li distingue;
 - la stringa 0 non li distingue, poiché li porta entrambi in q_1 ;
 - la stringa 1 non li distingue, poiché li porta entrambi in q_2 ;
 - dopo uno 0 o un 1, q_0 e q_2 confluiscono nello stesso stato e da lì in avanti il loro percorso è lo stesso. Dunque, non può esistere alcuna stringa che li distingua.

Esempio (dal libro)



- A e G sono distinti da 01 ma non da 0 né da ϵ
- A ed E sono indistinguibili (confluiscono dopo $1, 00$ e 01)

Algoritmo per trovare stati distinguibili

Input

Un DFA $A = (Q, \Sigma, \delta, q_0, F)$.

Output

L'insieme di tutte e sole le coppie di stati distinguibili di A .

Algoritmo

1. All'inizio nessuna coppia di stati è marcata come distinguibile.
2. Si marcano come distinguibili tutte le coppie $\{p, q\}$ in cui $p \in F$ e $q \notin F$.
3. Se esistono $p, q \in Q$ e $a \in \Sigma$ tali che $\{\delta(p, a), \delta(q, a)\}$ è marcata come distinguibile, si marca anche $\{p, q\}$ come distinguibile.
4. Si ripete il passo 3 fintantoché vengono marcate **nuove** coppie distinguibili.

Come eseguire l'algoritmo

- Si crea una **tabella triangolare** le cui righe sono etichettate con gli stati dal secondo all'ultimo e le cui colonne sono etichettate con gli stati dal primo al penultimo.
- Si marca con \checkmark ogni casella corrispondente a una coppia distinguibile (passi 2 e 3).
- Si itera il passo 3 considerando tutte le caselle non marcate fintantoché possibile.

Esempio

Seguono le tabelle corrispondenti ai passi 1–3 dell'algoritmo eseguito sul DFA della [slide 5](#).

Esempio

Seguono le tabelle corrispondenti ai passi 1–3 dell'algoritmo eseguito sul DFA della [slide 5](#).

B							
C							
D							
E							
F							
G							
H							
A	B	C	D	E	F	G	

B							
C	✓	✓					
D			✓				
E		✓					
F		✓					
G		✓					
H		✓					
A	B	C	D	E	F	G	

B	✓						
C	✓	✓					
D	✓	✓	✓				
E	✓	✓	✓	✓			
F	✓	✓	✓				✓
G	✓	✓	✓	✓	✓	✓	✓
H	✓		✓	✓	✓	✓	✓
A	B	C	D	E	F	G	

Note

- Iterando ancora una volta il passo 3 dell'algoritmo non si marcano altre coppie come distinguibili, dunque l'algoritmo termina.
- In generale possono servire più iterazioni del passo 3 prima che l'algoritmo termini.

L'indistinguibilità come equivalenza

Proposizione

L'indistinguibilità è una relazione di equivalenza, ovvero è riflessiva, simmetrica e transitiva.

Notazione

Fissato un DFA $A = (Q, \Sigma, \delta, q_0, F)$, introduciamo la seguente notazione:

- Indichiamo con \sim la **relazione di indistinguibilità** tra stati di A , ovvero $p \sim q$ se e solo se $\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F$ per ogni $w \in \Sigma^*$.
- Scriviamo $[p]$ per la **classe di equivalenza** di p , ovvero $[p] = \{q \in Q \mid p \sim q\}$.
- Scriviamo X/\sim per l'**insieme quoziante** di X rispetto a \sim , cioè $X/\sim = \{[p] \mid p \in X\}$.

Costruzione dell'automa minimo

Algoritmo

Dato un DFA $(Q, \Sigma, \delta, q_0, F)$, nel quale si assume di aver eliminato gli stati irraggiungibili dallo stato iniziale, l'**automa minimo** corrispondente è l'automa

$$(Q/\sim, \Sigma, \delta', [q_0], F/\sim)$$

in cui

$$\delta'([p], a) = [\delta(p, a)]$$

per ogni $p \in Q$ ed $a \in \Sigma$.

Teorema

Per ogni DFA A , non esiste un DFA equivalente il cui numero di stati è strettamente inferiore a quello dell'automa minimo corrispondente ad A costruito secondo l'algoritmo qui sopra.

Dimostrazione (facoltativa)

Si veda la Sezione 4.4.4 del libro.

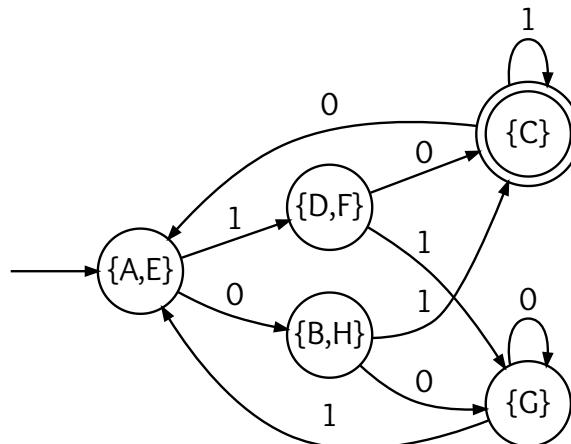
Esempio

Nell'automa della [slide 5](#) gli stati diversi ma indistinguibili sono

$$A \sim E \quad B \sim H \quad D \sim F$$

pertanto l'automa minimo, illustrato sotto, ha come insieme degli stati

$$\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$$



Equivalenza di automi

L'algoritmo **riempi-tabella** può essere usato per decidere se due automi sono **equivalenti**.

Input

Due DFA $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ e $A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ in cui possiamo assumere, senza perdere in generalità, che $Q_1 \cap Q_2 = \emptyset$.

Output

Vero se $L(A_1) = L(A_2)$ e falso altrimenti.

Algoritmo

1. Si crea l'unione dei due DFA $A = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_1 \cup F_2)$ dove

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{se } q \in Q_1 \\ \delta_2(q, a) & \text{se } q \in Q_2 \end{cases}$$

La scelta di eleggere q_1 come stato iniziale è fatta solo perché uno stato iniziale deve esserci.

2. Si esegue l'algoritmo riempi-tabella su A .
3. A_1 ed A_2 sono equivalenti se e solo se q_1 e q_2 sono indistinguibili in A .

Esercizi di minimizzazione

1. Costruire il DFA minimo equivalente ai seguenti automi (esercizi 4.4.1 e 4.4.2 del libro di testo):

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

	0	1
→A	B	E
B	C	F
*C	D	H
D	E	H
E	F	I
*F	G	B
G	H	B
H	I	C
*I	A	E

2. Costruire il DFA minimo equivalente all'espressione regolare a^*b^* .

Esercizi di equivalenza

1. Dimostrare che i seguenti automi sono equivalenti

