

Automati e Linguaggi Formali

Parte 14 – Linguaggi Decidibili

Davide Bresolin
Ultimo aggiornamento: 4 maggio 2022



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

- Studiare il potere degli algoritmi
- Capire quali problemi sono risolvibili da un algoritmo e quali no
- In questa lezione iniziamo considerando problemi **decidibili**

- 1** Problemi sui Linguaggi Regolari
- 2 Problemi per linguaggi Context-free
- 3 Relazioni tra classi di linguaggi

- **Problema dell'accettazione:** testare se un DFA accetta una stringa

$$A_{DFA} = \{\langle B, w \rangle \mid B \text{ è un DFA che accetta la stringa } w\}$$

- B accetta w se e solo se $\langle B, w \rangle$ appartiene ad A_{DFA}
- Mostrare che il linguaggio è **decidibile** equivale a mostrare che il problema computazionale è **decidibile**

Teorema: A_{DFA} è decidibile



Idea: definire una TM che decide A_{DFA}

Teorema: A_{DFA} è decidibile



Idea: definire una TM che decide A_{DFA}

$M =$ "Su input $\langle B, w \rangle$, dove B è un DFA e w una stringa:

- 1 Simula B su input w
- 2 Se la simulazione termina in uno stato finale, **accetta**. Se termina in uno stato non finale, **rifiuta**."

Teorema: A_{DFA} è decidibile



Idea: definire una TM che decide A_{DFA}

$M =$ "Su input $\langle B, w \rangle$, dove B è un DFA e w una stringa:

- 1 Simula B su input w
- 2 Se la simulazione termina in uno stato finale, **accetta**. Se termina in uno stato non finale, **rifiuta**."

Dimostrazione:

- la codifica di B è una lista dell componenti Q, Σ, δ, q_0 e F
- fare la simulazione è facile

Teorema: A_{NFA} è decidibile



$$A_{NFA} = \{\langle B, w \rangle \mid B \text{ è un } \varepsilon\text{-NFA che accetta la stringa } w\}$$

Idea: usiamo la TM M che decide A_{DFA} come subroutine

Teorema: A_{NFA} è decidibile



$$A_{NFA} = \{ \langle B, w \rangle \mid B \text{ è un } \varepsilon\text{-NFA che accetta la stringa } w \}$$

Idea: usiamo la TM M che decide A_{DFA} come subroutine

Dimostrazione:

N = "Su input $\langle B, w \rangle$, dove B è un ε -NFA e w una stringa:

- 1 Trasforma B in un DFA equivalente C usando la costruzione per sottoinsiemi
- 2 Esegui M con input $\langle C, w \rangle$
- 3 Se M accetta, **accetta**; altrimenti, **rifiuta**."

N è un decisore per A_{NFA} , quindi A_{NFA} è **decidibile**

Teorema: A_{REX} è decidibile



$A_{REX} = \{ \langle R, w \rangle \mid R \text{ è una espressione regolare che genera la stringa } w \}$

Idea: usiamo la TM N che decide A_{NFA} come subroutine

Teorema: A_{REG} è decidibile



$A_{REG} = \{ \langle R, w \rangle \mid R \text{ è una espressione regolare che genera la stringa } w \}$

Idea: usiamo la TM N che decide A_{NFA} come subroutine

Dimostrazione:

$P =$ "Su input $\langle R, w \rangle$, dove R è una espressione regolare e w una stringa:

- 1 Trasforma R in un ε -NFA equivalente C usando la procedura di conversione
- 2 Esegui N con input $\langle C, w \rangle$
- 3 Se N accetta, **accetta**; altrimenti, **rifiuta**."

P è un decisore per A_{REG} , quindi A_{REG} è **decidibile**

- ai fini della **decidibilità**, è equivalente dare in input alla TM un DFA, un ε -NFA o una espressione regolare
- la TM è in grado di convertire una codifica nell'altra
- **Ricorda**: mostrare che il linguaggio è **decidibile** equivale a mostrare che il problema computazionale è **decidibile**

- Negli esempi precedenti dovevamo decidere se una stringa appartenesse o no ad un linguaggio
- Ora vogliamo determinare se un automa finito accetta una **qualche** stringa

$$E_{DFA} = \{ \langle A \rangle \mid A \text{ è un DFA e } L(A) = \emptyset \}$$

- Puoi descrivere un algoritmo per eseguire questo test?

Dimostrazione: verifica se c'è uno stato finale che può essere raggiunto a partire dallo stato iniziale.

T = “Su input $\langle A \rangle$, la codifica di un DFA A :

- 1 **Marca** lo stato iniziale di A .
- 2 **Ripeti** la fase seguente fino a quando non vengono marcati nuovi stati:
- 3 **marca** ogni stato di A che ha una transizione proveniente da uno stato già marcato.
- 4 Se nessuno degli stati finali è marcato, **accetta**; altrimenti **rifiuta**.”

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

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

Idea:

- costruiamo un DFA C che accetta solo le stringhe che sono accettate da A o da B , ma non da entrambi
- se $L(A) = L(B)$ allora C non accetterà nulla
- il linguaggio di C è la **differenza simmetrica** di A e B

Dimostrazione:

- la **differenza simmetrica** di A e B è:

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$$

- i linguaggi regolari sono **chiusi** per unione, intersezione e complementazione
- $F =$ "Su input $\langle A, B \rangle$, dove A e B sono DFA:
 - 1 Costruisci il DFA C per differenza simmetrica
 - 2 Esegui T , la TM che decide E_{DFA} con input $\langle C \rangle$
 - 3 Se T accetta, **accetta**; altrimenti, **rifiuta**."

- 1 Problemi sui Linguaggi Regolari
- 2 Problemi per linguaggi Context-free
- 3 Relazioni tra classi di linguaggi

$$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ è una CFG che genera la stringa } w \}$$

Idea: costruiamo una TM che provi tutte le derivazioni di G per trovarne una che genera w

Perché questa strategia non funziona?



- Se la CFG è in forma normale di Chomsky, allora ogni derivazione di w è lunga **esattamente** $(2|w| - 1)$ passi
- Le TM possono convertire le grammatiche nella forma normale di Chomsky!

- Se la CFG è in forma normale di Chomsky, allora ogni derivazione di w è lunga **esattamente** $(2|w| - 1)$ passi
- Le TM possono convertire le grammatiche nella forma normale di Chomsky!

Dimostrazione:

$S =$ "Su input $\langle G, w \rangle$, dove G è una CFG e w una stringa:

- 1 Converti G in forma normale di Chomsky
- 2 Elenca tutte le derivazioni di $2|w| - 1$ passi. Se $|w| = 0$, elenca tutte le derivazioni di lunghezza 1
- 3 Se una delle derivazioni genera w , **accetta**; altrimenti **rifiuta**."

$$E_{CFG} = \{\langle G \rangle \mid A \text{ è una CFG ed } L(G) = \emptyset\}$$

- **Problema:** non possiamo usare S del teorema precedente.
Perché no?
- Bisogna procedere in modo diverso!

Idea: stabilisci per ogni variabile se è in grado di generare una stringa di terminali

$R =$ “Su input $\langle G \rangle$, la codifica di una CFG G :

- 1** **Marca** tutti i simboli terminali di G .
- 2** **Ripeti** la fase seguente fino a quando non vengono marcate nuove variabili:
 - 3** **marca** ogni variabile A tale che esiste una regola $A \rightarrow U_1 \dots U_k$ dove ogni simbolo $U_1 \dots U_k$ è già stato marcato.
- 4** Se la variabile iniziale non è marcata, **accetta**; altrimenti **rifiuta**.”

Teorema: EQ_{CFG} è decidibile



$$EQ_{CFG} = \{\langle G, H \rangle \mid G \text{ e } H \text{ sono CFG e } L(G) = L(H)\}$$

Idea:

- Usiamo la stessa tecnica di EQ_{DFA}
- Calcoliamo la **differenza simmetrica** di G e H per provare l'equivalenza

$$EQ_{CFG} = \{\langle G, H \rangle \mid G \text{ e } H \text{ sono CFG e } L(G) = L(H)\}$$

Idea:

- Usiamo la stessa tecnica di EQ_{DFA}
- Calcoliamo la **differenza simmetrica** di G e H per provare l'equivalenza

STOP!!!

- Le CFG non sono chiuse per complementazione ed intersezione!
- EQ_{CFG} non è decidibile!

- 1 Problemi sui Linguaggi Regolari
- 2 Problemi per linguaggi Context-free
- 3 Relazioni tra classi di linguaggi

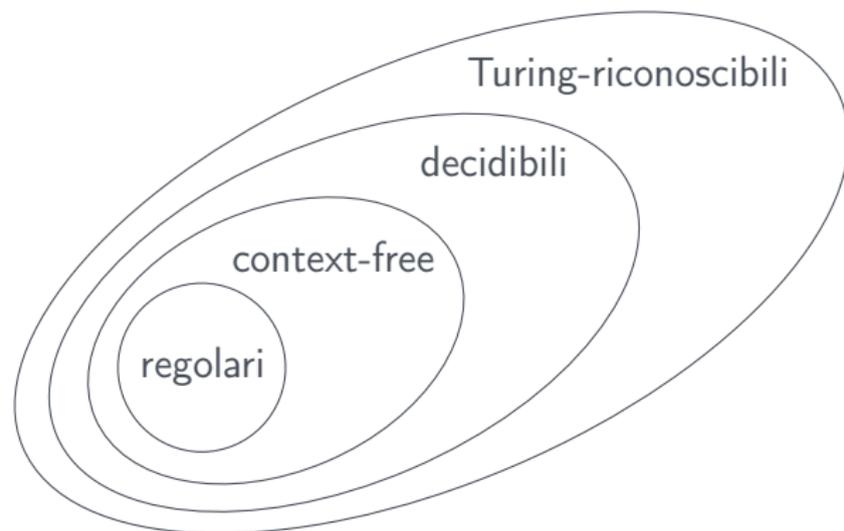
Domanda:

- è facile simulare la pila con una TM
- sappiamo che le TM nondeterministiche possono essere simulate da una TM deterministica

Non basta semplicemente simulare un PDA con una TM?

- Quali altre opzioni abbiamo?

- Dato un CFL L , sia G la grammatica per L
- Costruiamo la TM S che decide A_{CFG}
- La TM che decide L è:
 $M_G =$ "Su input w :
 - 1 Esegui la TM S con input $\langle G, w \rangle$
 - 2 Se S accetta, **accetta**; altrimenti, **rifiuta**



- Queste non sono solo classi di linguaggi, ma anche **classi di capacità computazionale**