DECIDIBILITÀ E INDECIDIBILITÀ

Il nostro obiettivo sarà studiare il potere computazionale delle macchine di Turing. Quindi analizzeremo i limiti della risoluzione dei problemi mediante algoritmi. Proveremo che esistono problemi che possono essere risolti mediante algoritmi e altri no.

DECIDIBILITÀ E INDECIDIBILITÀ

Problemi e Linguaggi - Problemi di decisione

Problemi di decisione

Un problema di decisione è un problema che ha come soluzione una risposta sì o no.

Esempi:

- Problema PRIMO: Dato un numero x intero e maggiore di uno, x è primo?

Problemi di decisione

Un problema di decisione è un problema che ha come soluzione una risposta sì o no.

Esempi:

- Problema PRIMO: Dato un numero x intero e maggiore di uno, x è primo?
- Problema CONNESSO: Dato un grafo G, G è connesso?

Problemi di decisione

Un problema di decisione è un problema che ha come soluzione una risposta sì o no.

Esempi:

- Problema PRIMO: Dato un numero x intero e maggiore di uno, x è primo?
- Problema CONNESSO: Dato un grafo G, G è connesso?
- Problema A_{DFA}: Dato un DFA $\mathcal{B}$ e una stringa w, $\mathcal{B}$ accetta w?

- Variabili Booleane: variabili che possono assumere valore VERO o FALSO. In genere rappresentiamo VERO con 1 e FALSO con 0.

- Variabili Booleane: variabili che possono assumere valore VERO o FALSO. In genere rappresentiamo VERO con 1 e FALSO con 0.
- Operazioni Booleane: $\vee$ (o *OR*), $\wedge$ (o *AND*), $\neg$ (o *NOT*)

Richiami di logica

- Variabili Booleane: variabili che possono assumere valore VERO o FALSO. In genere rappresentiamo VERO con 1 e FALSO con 0.
- Operazioni Booleane: $\vee$ (o *OR*), $\wedge$ (o *AND*), $\neg$ (o *NOT*)
- Denotiamo $\neg x$ con $\overline{x}$

Richiami di logica

- Variabili Booleane: variabili che possono assumere valore VERO o FALSO. In genere rappresentiamo VERO con 1 e FALSO con 0.
- Operazioni Booleane: $\vee$ (o *OR*), $\wedge$ (o *AND*), $\neg$ (o *NOT*)
- Denotiamo $\neg x$ con $\overline{x}$
- $0 \vee 0 = 0$, $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$

- Variabili Booleane: variabili che possono assumere valore VERO o FALSO. In genere rappresentiamo VERO con 1 e FALSO con 0.
- Operazioni Booleane: $\vee$ (o *OR*), $\wedge$ (o *AND*), $\neg$ (o *NOT*)
- Denotiamo $\neg x$ con $\overline{x}$
- $0 \vee 0 = 0$, $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$
- $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0$, $1 \wedge 1 = 1$

- Variabili Booleane: variabili che possono assumere valore VERO o FALSO. In genere rappresentiamo VERO con 1 e FALSO con 0.
- Operazioni Booleane: $\vee$ (o *OR*), $\wedge$ (o *AND*), $\neg$ (o *NOT*)
- Denotiamo $\neg x$ con $\overline{x}$
- $0 \vee 0 = 0$, $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$
- $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0$, $1 \wedge 1 = 1$
- $\overline{0} = 1$, $\overline{1} = 0$

Definizione

Dato un insieme di variabili booleane X, le formule booleane (o espressioni booleane) su X sono definite induttivamente come segue:

- *le costanti* 0, 1 *e le variabili (in forma diretta o complementata)* x, $\overline{x}$, *con* $x \in X$, *sono formule booleane.*

Definizione

Dato un insieme di variabili booleane X, le formule booleane (o espressioni booleane) su X sono definite induttivamente come segue:

- *le costanti* 0, 1 *e le variabili (in forma diretta o complementata)* x, $\overline{x}$, *con* $x \in X$, *sono formule booleane.*
- *Se* ϕ, ϕ_1, ϕ_2 *sono formule booleane allora* $(\phi_1 \vee \phi_2)$, $(\phi_1 \wedge \phi_2)$, $\overline{\phi}$ *sono formule booleane.*

Una formula booleana ϕ è soddisfacibile se esiste un insieme di valori 0 o 1 per le variabili di ϕ (o assegnamento) che renda la formula uguale a 1 (assegnamento di soddisfacibilità). Diremo che tale assegnamento soddisfa ϕ o anche che rende vera ϕ.

Una formula booleana ϕ è soddisfacibile se esiste un insieme di valori 0 o 1 per le variabili di ϕ (o assegnamento) che renda la formula uguale a 1 (assegnamento di soddisfacibilità). Diremo che tale assegnamento soddisfa ϕ o anche che rende vera ϕ.

Esempi:

Una formula booleana ϕ è soddisfacibile se esiste un insieme di valori 0 o 1 per le variabili di ϕ (o assegnamento) che renda la formula uguale a 1 (assegnamento di soddisfacibilità). Diremo che tale assegnamento soddisfa ϕ o anche che rende vera ϕ.

Esempi:

- $\phi_1 = (\overline{x} \vee y) \wedge (x \vee \overline{z})$ è soddisfacibile (assegnamento: $x = 0$, $y = 1$, $z = 0$),

Una formula booleana ϕ è soddisfacibile se esiste un insieme di valori 0 o 1 per le variabili di ϕ (o assegnamento) che renda la formula uguale a 1 (assegnamento di soddisfacibilità). Diremo che tale assegnamento soddisfa ϕ o anche che rende vera ϕ.

Esempi:

- $\phi_1 = (\overline{x} \vee y) \wedge (x \vee \overline{z})$ è soddisfacibile (assegnamento: $x = 0$, $y = 1$, $z = 0$),
- $\phi_2 = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ è soddisfacibile,

Una formula booleana ϕ è soddisfacibile se esiste un insieme di valori 0 o 1 per le variabili di ϕ (o assegnamento) che renda la formula uguale a 1 (assegnamento di soddisfacibilità). Diremo che tale assegnamento soddisfa ϕ o anche che rende vera ϕ.

Esempi:

- $\phi_1 = (\overline{x} \vee y) \wedge (x \vee \overline{z})$ è soddisfacibile (assegnamento: $x = 0$, $y = 1$, $z = 0$),
- $\phi_2 = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ è soddisfacibile,
- $\phi_3 = (\overline{x} \vee x) \wedge (y \vee \overline{y})$ è soddisfacibile (per qualunque assegnamento di valori delle variabili),

Una formula booleana ϕ è soddisfacibile se esiste un insieme di valori 0 o 1 per le variabili di ϕ (o assegnamento) che renda la formula uguale a 1 (assegnamento di soddisfacibilità). Diremo che tale assegnamento soddisfa ϕ o anche che rende vera ϕ.

Esempi:

- $\phi_1 = (\overline{x} \vee y) \wedge (x \vee \overline{z})$ è soddisfacibile (assegnamento: $x = 0$, $y = 1$, $z = 0$),
- $\phi_2 = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ è soddisfacibile,
- $\phi_3 = (\overline{x} \vee x) \wedge (y \vee \overline{y})$ è soddisfacibile (per qualunque assegnamento di valori delle variabili),
- $\phi_4 = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ non è soddisfacibile.

Una formula booleana ϕ è soddisfacibile se esiste un assegnamento di valori di verità 0 o 1 alle variabili di ϕ che soddisfa ϕ, cioè che renda la formula uguale a 1.

Problemi di decisione

Una formula booleana ϕ è soddisfacibile se esiste un assegnamento di valori di verità 0 o 1 alle variabili di ϕ che soddisfa ϕ, cioè che renda la formula uguale a 1.

- Problema SAT: Data una formula booleana ϕ, ϕ è soddisfacibile?

Problemi di decisione

Una formula booleana ϕ è soddisfacibile se esiste un assegnamento di valori di verità 0 o 1 alle variabili di ϕ che soddisfa ϕ, cioè che renda la formula uguale a 1.

- Problema SAT: Data una formula booleana ϕ, ϕ è soddisfacibile?

Un cammino Hamiltoniano in un grafo orientato è un cammino (orientato) che passa per ogni vertice del grafo una e una sola volta.

Problemi di decisione

Una formula booleana ϕ è soddisfacibile se esiste un assegnamento di valori di verità 0 o 1 alle variabili di ϕ che soddisfa ϕ, cioè che renda la formula uguale a 1.

- Problema SAT: Data una formula booleana ϕ, ϕ è soddisfacibile?

Un cammino Hamiltoniano in un grafo orientato è un cammino (orientato) che passa per ogni vertice del grafo una e una sola volta.

- Problema HAMPATH: Dato un grafo orientato G e due vertici s e t, esiste un cammino hamiltoniano nel grafo da s a t?

Problemi di decisione
istanze

Un problema di decisione considera elementi di un insieme.

Problemi di decisione
istanze

Un problema di decisione considera elementi di un insieme.

Tali elementi sono anche chiamati le **istanze** del problema.

Problemi di decisione
istanze

Un problema di decisione considera elementi di un insieme.

Tali elementi sono anche chiamati le **istanze** del problema.

Quindi un istanza di un problema è un particolare input per quel problema.

Problemi di decisione
istanze

Un problema di decisione considera elementi di un insieme.

Tali elementi sono anche chiamati le **istanze** del problema.

Quindi un istanza di un problema è un particolare input per quel problema.

- Esempio: 3, 4, 6 sono istanze per il problema PRIMO.

Problemi di decisione
istanze

Un problema di decisione considera elementi di un insieme.

Tali elementi sono anche chiamati le **istanze** del problema.

Quindi un istanza di un problema è un particolare input per quel problema.

- Esempio: 3, 4, 6 sono istanze per il problema PRIMO.
- Esempio: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2)$ è un istanza per il problema SAT.

Problemi di decisione
istanze sì e istanze no

L'insieme delle istanze (per un problema di decisione) è unione del sottoinsieme delle **istanze con risposta sì** e del sottoinsieme delle **istanze con risposta no**.

Problemi di decisione
istanze sì e istanze no

L'insieme delle istanze (per un problema di decisione) è unione del sottoinsieme delle **istanze con risposta sì** e del sottoinsieme delle **istanze con risposta no**.

- Esempio: 3 è un'istanza sì per il problema PRIMO, 4 e 6 sono istanze no per il problema PRIMO.

Problemi di decisione
istanze sì e istanze no

L'insieme delle istanze (per un problema di decisione) è unione del sottoinsieme delle **istanze con risposta sì** e del sottoinsieme delle **istanze con risposta no**.

- Esempio: 3 è un'istanza sì per il problema PRIMO, 4 e 6 sono istanze no per il problema PRIMO.
- Esempio: $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2)$ è un istanza sì per il problema SAT (prendere $x_1 = x_2 = 1$).

$$(x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$$

è un istanza no per il problema SAT.

Problemi di decisione
istanze sì e istanze no

- Nel problema PRIMO le istanze sono i numeri x interi e maggiori di uno.
  L'insieme $\{n \in \mathbb{N} \mid n > 1\}$ di tali numeri è unione dell'insieme dei numeri primi (istanze sì) e dell'insieme dei numeri non primi o composti (istanze no).

Problemi di decisione
istanze sì e istanze no

- Nel problema PRIMO le istanze sono i numeri x interi e maggiori di uno.
  L'insieme $\{n \in \mathbb{N} \mid n > 1\}$ di tali numeri è unione dell'insieme dei numeri primi (istanze sì) e dell'insieme dei numeri non primi o composti (istanze no).
- Nel problema CONNESSO le istanze sono grafi G.
  L'insieme dei grafi è unione dell'insieme dei grafi connessi (istanze sì) e dell'insieme dei grafi non connessi (istanze no).

Problemi di decisione
istanze sì e istanze no

- Nel problema PRIMO le istanze sono i numeri x interi e maggiori di uno.
  L'insieme $\{n \in \mathbb{N} \mid n > 1\}$ di tali numeri è unione dell'insieme dei numeri primi (istanze sì) e dell'insieme dei numeri non primi o composti (istanze no).
- Nel problema CONNESSO le istanze sono grafi G.
  L'insieme dei grafi è unione dell'insieme dei grafi connessi (istanze sì) e dell'insieme dei grafi non connessi (istanze no).
- Nel problema A_{DFA} le istanze sono coppie $(\mathcal{B}, w)$ costituite da un DFA $\mathcal{B}$ e da una stringa w.
  L'insieme di tali coppie $(\mathcal{B}, w)$ è unione dell'insieme delle $(\mathcal{B}, w)$ con $w \in L(\mathcal{B})$ (istanze sì) e dell'insieme delle $(\mathcal{B}, w)$ con $w \notin L(\mathcal{B})$ (istanze no).

Problemi di decisione e ricerca di una soluzione

In realtà in generale siamo interessati a trovare una soluzione piuttosto che a sapere se c'è una soluzione.

Problemi di decisione e ricerca di una soluzione

In realtà in generale siamo interessati a trovare una soluzione piuttosto che a sapere se c'è una soluzione.

Chiamiamo problemi di ricerca quelli per i quali cerchiamo una soluzione (se esiste).

Problemi di decisione e ricerca di una soluzione

- Esempio
  RSAT: Data una formula booleana ϕ, fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ.

Problemi di decisione e ricerca di una soluzione

- Esempio
  RSAT: Data una formula booleana ϕ, fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ.

- Esempio
  RHAMPATH: Dato un grafo orientato G e due vertici s e t, fornire, se esiste, un cammino hamiltoniano nel grafo da s a t.

Problemi di decisione e ricerca di una soluzione

Dato un problema di ricerca possiamo in genere usare come sottoprogramma un algoritmo per il corrispondente problema di decisione, se tale algoritmo esiste (altrimenti né l'uno né l'altro è algoritmicamente risolubile).

Questo giustifica la concentrazione sui problemi di decisione.

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

- Usiamo **AI** per stabilire se ϕ è soddisfacibile.

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

- Usiamo **AI** per stabilire se ϕ è soddisfacibile.
- Se ϕ non è soddisfacibile abbiamo una risposta al nostro problema di ricerca: l'assegnamento non esiste. Se ϕ è soddisfacibile

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

- Usiamo **AI** per stabilire se ϕ è soddisfacibile.
- Se ϕ non è soddisfacibile abbiamo una risposta al nostro problema di ricerca: l'assegnamento non esiste. Se ϕ è soddisfacibile
  1. Assegniamo a una delle variabili x il valore 1.

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

- Usiamo **AI** per stabilire se ϕ è soddisfacibile.
- Se ϕ non è soddisfacibile abbiamo una risposta al nostro problema di ricerca: l'assegnamento non esiste. Se ϕ è soddisfacibile
  1. Assegniamo a una delle variabili x il valore 1.
  2. Eseguiamo **AI** sulla nuova formula ottenuta.

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

- Usiamo **AI** per stabilire se ϕ è soddisfacibile.
- Se ϕ non è soddisfacibile abbiamo una risposta al nostro problema di ricerca: l'assegnamento non esiste. Se ϕ è soddisfacibile
  1. Assegniamo a una delle variabili x il valore 1.
  2. Eseguiamo **AI** sulla nuova formula ottenuta.
  3. Se tale formula non è soddisfacibile, assegniamo a x il valore 0. La formula ottenuta sarà ora soddisfacibile perché se ϕ è soddisfacibile, uno dei due valori è quello giusto.

Problemi di decisione e ricerca di una soluzione

Esempio: Data una formula booleana ϕ, vogliamo fornire, se esiste, un assegnamento di valori di verità che soddisfa ϕ. Disponiamo di un algoritmo **AI** che risolve SAT.

- Usiamo **AI** per stabilire se ϕ è soddisfacibile.
- Se ϕ non è soddisfacibile abbiamo una risposta al nostro problema di ricerca: l'assegnamento non esiste. Se ϕ è soddisfacibile
  1. Assegniamo a una delle variabili x il valore 1.
  2. Eseguiamo **AI** sulla nuova formula ottenuta.
  3. Se tale formula non è soddisfacibile, assegniamo a x il valore 0. La formula ottenuta sarà ora soddisfacibile perché se ϕ è soddisfacibile, uno dei due valori è quello giusto.
  4. Chiamiamo ϕ' la formula ottenuta da ϕ assegnando in essa a x il "giusto" valore. Se in ϕ' vi sono ancora variabili, riapplichiamo la procedura dal passo 1 a ϕ'.

DECIDIBILITÀ E INDECIDIBILITÀ

Linguaggio associato a un problema di decisione

Livelli di descrizione di una macchina di Turing

Il nostro obiettivo è analizzare i limiti della risoluzione dei problemi mediante algoritmi, cioè mostrare l'esistenza di problemi non algoritmicamente risolubili.

Livelli di descrizione di una macchina di Turing

Il nostro obiettivo è analizzare i limiti della risoluzione dei problemi mediante algoritmi, cioè mostrare l'esistenza di problemi non algoritmicamente risolubili.

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Livelli di descrizione di una macchina di Turing

Il nostro obiettivo è analizzare i limiti della risoluzione dei problemi mediante algoritmi, cioè mostrare l'esistenza di problemi non algoritmicamente risolubili.

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Come descriveremo le macchine di Turing?
Ci sono tre possibili livelli di descrizione:

Livelli di descrizione di una macchina di Turing

Il nostro obiettivo è analizzare i limiti della risoluzione dei problemi mediante algoritmi, cioè mostrare l'esistenza di problemi non algoritmicamente risolubili.

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Come descriveremo le macchine di Turing?
Ci sono tre possibili livelli di descrizione:

- la descrizione formale

Livelli di descrizione di una macchina di Turing

Il nostro obiettivo è analizzare i limiti della risoluzione dei problemi mediante algoritmi, cioè mostrare l'esistenza di problemi non algoritmicamente risolubili.

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Come descriveremo le macchine di Turing?
Ci sono tre possibili livelli di descrizione:

- la descrizione formale
- la descrizione implementativa

Livelli di descrizione di una macchina di Turing

Il nostro obiettivo è analizzare i limiti della risoluzione dei problemi mediante algoritmi, cioè mostrare l'esistenza di problemi non algoritmicamente risolubili.

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Come descriveremo le macchine di Turing?
Ci sono tre possibili livelli di descrizione:

- la descrizione formale
- la descrizione implementativa
- la descrizione ad alto livello

Linguaggio associato a un problema di decisione

Abbiamo detto che assumeremo che “algoritmo” sia sinonimo di “macchina di Turing”.

Linguaggio associato a un problema di decisione

Abbiamo detto che assumeremo che “algoritmo” sia sinonimo di “macchina di Turing”.

Useremo descrizioni ad alto livello delle macchine di Turing.

Linguaggio associato a un problema di decisione

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Useremo descrizioni ad alto livello delle macchine di Turing.

L'input di una macchina di Turing è una stringa. Se vogliamo dare in input altri oggetti, questi devono essere codificati come stringhe.

Linguaggio associato a un problema di decisione

Abbiamo detto che assumeremo che "algoritmo" sia sinonimo di "macchina di Turing".

Useremo descrizioni ad alto livello delle macchine di Turing.

L'input di una macchina di Turing è una stringa. Se vogliamo dare in input altri oggetti, questi devono essere codificati come stringhe.

Quindi le istanze devono essere "rappresentate" come stringhe su un alfabeto.

Codifiche

- Le istanze del nostro problema saranno rappresentate con stringhe su un alfabeto Σ.

Codifiche

- Le istanze del nostro problema saranno rappresentate con stringhe su un alfabeto Σ.
- La corrispondenza che a una istanza associa una stringa deve essere una **codifica**, cioè deve rappresentare univocamente l'istanza (e solo quella istanza).

Codifiche

- Le istanze del nostro problema saranno rappresentate con stringhe su un alfabeto Σ.
- La corrispondenza che a una istanza associa una stringa deve essere una **codifica**, cioè deve rappresentare univocamente l'istanza (e solo quella istanza).
- Nel seguito non definiremo nei dettagli le codifiche degli oggetti, daremo solo qualche esempio.

Codifiche

- Le istanze del nostro problema saranno rappresentate con stringhe su un alfabeto Σ.
- La corrispondenza che a una istanza associa una stringa deve essere una **codifica**, cioè deve rappresentare univocamente l'istanza (e solo quella istanza).
- Nel seguito non definiremo nei dettagli le codifiche degli oggetti, daremo solo qualche esempio.
- Useremo

$$\langle \mathcal{O} \rangle$$

per denotare una stringa che codifica l'oggetto $\mathcal{O}$, useremo

Codifiche

- Le istanze del nostro problema saranno rappresentate con stringhe su un alfabeto Σ.
- La corrispondenza che a una istanza associa una stringa deve essere una **codifica**, cioè deve rappresentare univocamente l'istanza (e solo quella istanza).
- Nel seguito non definiremo nei dettagli le codifiche degli oggetti, daremo solo qualche esempio.
- Useremo

$$\langle \mathcal{O} \rangle$$

per denotare una stringa che codifica l'oggetto $\mathcal{O}$, useremo

$$\langle \mathcal{O}_1, \ldots, \mathcal{O}_k \rangle$$

per denotare una stringa che codifica gli oggetti $\mathcal{O}_1, \ldots, \mathcal{O}_k$.

Codifiche

Va bene una codifica qualsiasi delle istanze?

Codifiche

Va bene una codifica qualsiasi delle istanze?

In questo ambito sì, va bene una codifica qualsiasi delle istanze, a condizione che

Va bene una codifica qualsiasi delle istanze?

In questo ambito sì, va bene una codifica qualsiasi delle istanze, a condizione che

- la codifica sia definita mediante un algoritmo informale,

Codifiche

Va bene una codifica qualsiasi delle istanze?

In questo ambito sì, va bene una codifica qualsiasi delle istanze, a condizione che

- la codifica sia definita mediante un algoritmo informale,
- sia possibile discriminare le stringhe codifiche di istanze da quelle che non lo sono mediante un algoritmo informale,

Codifiche

Va bene una codifica qualsiasi delle istanze?

In questo ambito sì, va bene una codifica qualsiasi delle istanze, a condizione che

- la codifica sia definita mediante un algoritmo informale,
- sia possibile discriminare le stringhe codifiche di istanze da quelle che non lo sono mediante un algoritmo informale,
- sia possibile, mediante un algoritmo informale, risalire dalla codifica all'unica istanza che la codifica rappresenta.

Va bene una codifica qualsiasi delle istanze?

In questo ambito sì, va bene una codifica qualsiasi delle istanze, a condizione che

- la codifica sia definita mediante un algoritmo informale,
- sia possibile discriminare le stringhe codifiche di istanze da quelle che non lo sono mediante un algoritmo informale,
- sia possibile, mediante un algoritmo informale, risalire dalla codifica all'unica istanza che la codifica rappresenta.

Esempio. Possiamo scegliere come codifica di un numero intero non negativo la sua rappresentazione binaria. Con questa scelta avremo ad esempio

$$\langle 4 \rangle = 100, \quad \langle 7 \rangle = 111$$

Poi vedremo come nella teoria della complessità assuma rilevanza anche considerare codifiche **non "prolisse"** cioè tali che non vi siano istanze la cui rappresentazione sia artificiosamente lunga.

Poi vedremo come nella teoria della complessità assuma rilevanza anche considerare codifiche **non "prolisse"** cioè tali che non vi siano istanze la cui rappresentazione sia artificiosamente lunga.

Esempio: considerare codifiche in base $k \geq 2$ dei numeri (cioè escludere la rappresentazione unaria dei numeri),

Poi vedremo come nella teoria della complessità assuma rilevanza anche considerare codifiche **non "prolisse"** cioè tali che non vi siano istanze la cui rappresentazione sia artificiosamente lunga.

Esempio: considerare codifiche in base $k \geq 2$ dei numeri (cioè escludere la rappresentazione unaria dei numeri),

grafi come coppie di insiemi (di nodi e archi) o mediante la matrice di adiacenza,

Poi vedremo come nella teoria della complessità assuma rilevanza anche considerare codifiche **non "prolisse"** cioè tali che non vi siano istanze la cui rappresentazione sia artificiosamente lunga.

Esempio: considerare codifiche in base $k \geq 2$ dei numeri (cioè escludere la rappresentazione unaria dei numeri),

grafi come coppie di insiemi (di nodi e archi) o mediante la matrice di adiacenza,

insiemi, relazioni, funzioni mediante enumerazione delle codifiche dei relativi elementi....

- Esempio:
  Problema CONNESSO: Dato un grafo G, G è connesso?

- Esempio:
  Problema CONNESSO: Dato un grafo G, G è connesso?
- Le istanze in questo problema sono i grafi.

Codifiche

- Esempio:
  Problema CONNESSO: Dato un grafo G, G è connesso?
- Le istanze in questo problema sono i grafi.
- $\langle G \rangle$ rappresenta una codifica di G.

Codifiche

Come possiamo codificare un grafo G mediante una stringa su un alfabeto Σ?

Codifiche

Come possiamo codificare un grafo G mediante una stringa su un alfabeto Σ?

Una possibile codifica è illustrata su un esempio.

Come possiamo codificare un grafo G mediante una stringa su un alfabeto Σ?

Una possibile codifica è illustrata su un esempio.

Consideriamo il grafo

$$G = (\{1, 2, 3\}, \{(1, 2), (2, 3), (3, 1)\})$$

Come possiamo codificare un grafo G mediante una stringa su un alfabeto Σ?

Una possibile codifica è illustrata su un esempio.

Consideriamo il grafo

$$G = (\{1, 2, 3\}, \{(1, 2), (2, 3), (3, 1)\})$$

Possiamo prendere $\Sigma = \{0, 1, (,), \#\}$.
e associare a G la stringa:

$$\langle G \rangle = (1\#10\#11)\#((1\#10)\#(10\#11)\#(11\#1)).$$

È possibile codificare una MT M con una stringa.

È possibile codificare una MT M con una stringa.

È possibile anche codificare una MT M e una stringa w con una stringa.

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

- i simboli dell'alfabeto degli input,

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

- i simboli dell'alfabeto degli input,
- i simboli dell'alfabeto di nastro,

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

- i simboli dell'alfabeto degli input,
- i simboli dell'alfabeto di nastro,
- gli stati,

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

- i simboli dell'alfabeto degli input,
- i simboli dell'alfabeto di nastro,
- gli stati,
- i possibili movimenti L, R (ed eventualmente S) della testina,

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

- i simboli dell'alfabeto degli input,
- i simboli dell'alfabeto di nastro,
- gli stati,
- i possibili movimenti L, R (ed eventualmente S) della testina,
- i valori della funzione di transizione,

Codifica di una MdT

In generale per codificare una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ occorre stabilire come codificare

- i simboli dell'alfabeto degli input,
- i simboli dell'alfabeto di nastro,
- gli stati,
- i possibili movimenti L, R (ed eventualmente S) della testina,
- i valori della funzione di transizione,
- lo stato iniziale q_0 e gli stati q_{accept}, q_{reject}.

Codifica di una MdT - Approfondimento

Una possibile codifica di una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ mediante una stringa su $\{0, 1\}$

- Fissiamo un alfabeto infinito universale $\Sigma_U = \{a_1, a_2, \ldots\}$ e assumiamo che tutti i simboli input e di nastro siano estratti da Σ_U.

Codifica di una MdT - Approfondimento

Una possibile codifica di una macchina di Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ mediante una stringa su $\{0, 1\}$

- Fissiamo un alfabeto infinito universale $\Sigma_U = \{a_1, a_2, \ldots\}$ e assumiamo che tutti i simboli input e di nastro siano estratti da Σ_U.

- Fissiamo un insieme infinito universale di stati $Q_U = \{q_0, q_1, \ldots\}$ e assumiamo che tutte le TM usano nomi di stati scelti da Q_U.

Codifica di una MdT - Approfondimento

- Codifichiamo un elemento $a_i \in \Sigma_U$ con la stringa $e(a_i) = 0^{i+1}$

Codifica di una MdT - Approfondimento

- Codifichiamo un elemento $a_i \in \Sigma_U$ con la stringa $e(a_i) = 0^{i+1}$
  Codifichiamo il simbolo $\sqcup$ con la stringa $e(\sqcup) = 0$

Codifica di una MdT - Approfondimento

- Codifichiamo un elemento $a_i \in \Sigma_U$ con la stringa $e(a_i) = 0^{i+1}$
  Codifichiamo il simbolo $\sqcup$ con la stringa $e(\sqcup) = 0$

- Codifichiamo un alfabeto $\Delta = \{b_1, b_2, \ldots, b_r\}$ mediante la stringa

$$e(\Delta) = 111e(b_1)1e(b_2)1 \cdots 1e(b_r)111$$

  Quindi $e(\Sigma)$ ed $e(\Gamma)$ sono definiti.

Codifica di una MdT - Approfondimento

- Codifichiamo un elemento $q_i \in Q_U$ con la stringa $e(q_i) = 0^{i+1}$

Codifica di una MdT - Approfondimento

- Codifichiamo un elemento $q_i \in Q_U$ con la stringa $e(q_i) = 0^{i+1}$
- Codifichiamo i movimenti della testina con

$$e(L) = 0, \quad e(R) = 00 \quad \text{(ed eventualmente } e(S) = 000\text{)}$$

Codifica di una MdT - Approfondimento

- Codifichiamo un elemento $q_i \in Q_U$ con la stringa $e(q_i) = 0^{i+1}$
- Codifichiamo i movimenti della testina con

$$e(L) = 0, \quad e(R) = 00 \quad \text{(ed eventualmente } e(S) = 000\text{)}$$

- Codifichiamo una transizione m di una TM, ad esempio $\delta(q, a) = (p, b, D)$, con la stringa

$$e(m) = 11e(q)1e(a)1e(p)1e(b)1e(D)11$$

Codifica di una MdT - Approfondimento

- Infine codifichiamo l'intera TM
  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ con transizioni
  $m_1, m_2, \ldots m_t$ mediante la stringa

$$\langle M \rangle =$$

$$11111e(q_0)1e(q_{accept})1e(q_{reject})1e(\Sigma)1e(\Gamma)1e(m_1)1 \cdots 1e(m_t)11111$$

Codifica di una MdT - Approfondimento

- Infine codifichiamo l'intera TM
  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ con transizioni
  $m_1, m_2, \ldots m_t$ mediante la stringa

$$\langle M \rangle =$$

$$11111e(q_0)1e(q_{accept})1e(q_{reject})1e(\Sigma)1e(\Gamma)1e(m_1)1\cdots 1e(m_t)11111$$

- Codifichiamo una stringa $w = b_1 b_2 \cdots b_r$ mediante
  $e(w) = 11e(b_1)1e(b_2)1\cdots 1e(b_r)11$

Codifica di una MdT - Approfondimento

- Infine codifichiamo l'intera TM
  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ con transizioni
  $m_1, m_2, \ldots m_t$ mediante la stringa

$$\langle M \rangle =$$

$$11111e(q_0)1e(q_{accept})1e(q_{reject})1e(\Sigma)1e(\Gamma)1e(m_1)1 \cdots 1e(m_t)11111$$

- Codifichiamo una stringa $w = b_1 b_2 \cdots b_r$ mediante
  $e(w) = 11e(b_1)1e(b_2)1 \cdots 1e(b_r)11$
- Codifichiamo una MT M e una stringa w con la stringa

$$\langle M, w \rangle = e(M)e(w)$$

Linguaggio associato a un problema di decisione

- Mentre l'insieme delle istanze si divide in due sottoinsiemi (l'insieme delle istanze sì e quello delle istanze no), l'insieme delle stringhe su Σ si divide in **tre** sottoinsiemi:

Linguaggio associato a un problema di decisione

- Mentre l'insieme delle istanze si divide in due sottoinsiemi (l'insieme delle istanze sì e quello delle istanze no), l'insieme delle stringhe su Σ si divide in **tre** sottoinsiemi:

  1. L'insieme delle stringhe w che codificano istanze con **risposta sì**.

Linguaggio associato a un problema di decisione

- Mentre l'insieme delle istanze si divide in due sottoinsiemi (l'insieme delle istanze sì e quello delle istanze no), l'insieme delle stringhe su Σ si divide in **tre** sottoinsiemi:

  1. L'insieme delle stringhe w che codificano istanze con **risposta sì**.
  2. L'insieme delle stringhe w che codificano istanze con **risposta no**.

Linguaggio associato a un problema di decisione

- Mentre l'insieme delle istanze si divide in due sottoinsiemi (l'insieme delle istanze sì e quello delle istanze no), l'insieme delle stringhe su Σ si divide in **tre** sottoinsiemi:

  1. L'insieme delle stringhe w che codificano istanze con **risposta sì**.
  2. L'insieme delle stringhe w che codificano istanze con **risposta no**.
  3. L'insieme delle stringhe w che **non sono codifiche di istanze**.

Linguaggio associato a un problema di decisione

- Mentre l'insieme delle istanze si divide in due sottoinsiemi (l'insieme delle istanze sì e quello delle istanze no), l'insieme delle stringhe su Σ si divide in **tre** sottoinsiemi:

  1. L'insieme delle stringhe w che codificano istanze con **risposta sì**.
  2. L'insieme delle stringhe w che codificano istanze con **risposta no**.
  3. L'insieme delle stringhe w che **non sono codifiche di istanze**.

- Il linguaggio L **associato** a un problema di decisione $\mathbb{P}$ è il linguaggio delle **codifiche** delle istanze che hanno **risposta sì**.

Linguaggio associato a un problema di decisione

- Il linguaggio L **associato** a un problema di decisione $\mathbb{P}$ è il linguaggio delle **codifiche** delle istanze che hanno **risposta sì**.

Linguaggio associato a un problema di decisione

- Il linguaggio L **associato** a un problema di decisione $\mathbb{P}$ è il linguaggio delle **codifiche** delle istanze che hanno **risposta sì**.
- Se esiste una macchina di Turing che decide L il problema viene detto decidibile. Altrimenti il problema viene detto indecidibile.

Linguaggio associato a un problema di decisione

- Il linguaggio L **associato** a un problema di decisione $\mathbb{P}$ è il linguaggio delle **codifiche** delle istanze che hanno **risposta sì**.
- Se esiste una macchina di Turing che decide L il problema viene detto decidibile. Altrimenti il problema viene detto indecidibile.
- Se esiste una macchina di Turing che riconosce L il problema viene detto semidecidibile.

Linguaggio associato a un problema di decisione

- Il linguaggio L **associato** a un problema di decisione $\mathbb{P}$ è il linguaggio delle **codifiche** delle istanze che hanno **risposta sì**.
- Se esiste una macchina di Turing che decide L il problema viene detto decidibile. Altrimenti il problema viene detto indecidibile.
- Se esiste una macchina di Turing che riconosce L il problema viene detto semidecidibile.
- In questo modo esprimiamo un problema computazionale come un problema di riconoscimento di un linguaggio. La macchina corrisponde a un algoritmo per il problema.

Linguaggio associato a un problema di decisione

Per esempio, il linguaggio associato al problema "CONNESSO" è

$$A = \{\langle G \rangle \mid G \text{ è un grafo connesso}\}$$

Linguaggio associato a un problema di decisione

Per esempio, il linguaggio associato al problema "CONNESSO" è

$$A = \{\langle G \rangle \mid G \text{ è un grafo connesso}\}$$

dove $\langle G \rangle$ denota una codifica di G mediante una stringa su un alfabeto Σ.

Linguaggio associato a un problema di decisione

Per esempio, il linguaggio associato al problema "CONNESSO" è

$$A = \{\langle G \rangle \mid G \text{ è un grafo connesso}\}$$

dove $\langle G \rangle$ denota una codifica di G mediante una stringa su un alfabeto Σ.

Risolvere CONNESSO equivale a trovare una macchina di Turing che decida A.

Linguaggio associato a un problema di decisione

Sia $G = (V, E)$ un grafo non orientato, con insieme V di nodi e insieme E di archi. Un sottoinsieme V' di nodi di G è un *independent set* in G se per ogni u, v in V', la coppia (u, v) non è un arco, cioè u e v non sono adiacenti.

Linguaggio associato a un problema di decisione

Sia $G = (V, E)$ un grafo non orientato, con insieme V di nodi e insieme E di archi. Un sottoinsieme V' di nodi di G è un *independent set* in G se per ogni u, v in V', la coppia (u, v) non è un arco, cioè u e v non sono adiacenti.

Definire il linguaggio *INDEPENDENT-SET* associato al seguente problema di decisione:

Sia G un grafo non orientato e k un intero positivo. G ha un independent set di cardinalità k?

Linguaggio associato a un problema di decisione

Sia $G = (V, E)$ un grafo non orientato, con insieme V di nodi e insieme E di archi. Un sottoinsieme V' di nodi di G è un *independent set* in G se per ogni u, v in V', la coppia (u, v) non è un arco, cioè u e v non sono adiacenti.

Definire il linguaggio *INDEPENDENT-SET* associato al seguente problema di decisione:

Sia G un grafo non orientato e k un intero positivo. G ha un independent set di cardinalità k?

INDEPENDENT-SET $= \{\langle G, k\rangle \mid G$ è un grafo non orientato, k è un intero positivo e G ha un independent set di cardinalità $k\}$

Linguaggio associato a un problema di decisione

Definire il linguaggio E_{TM} associato al seguente problema di decisione:

Input: *M macchina di Turing.*
Domanda: *Il linguaggio $L(M)$ riconosciuto da M è vuoto?*

Linguaggio associato a un problema di decisione

Definire il linguaggio E_{TM} associato al seguente problema di decisione:

Input: *M macchina di Turing.*
Domanda: *Il linguaggio $L(M)$ riconosciuto da M è vuoto?*

$E_{TM} = \{\langle M \rangle \mid M \text{ è una MdT tale che } L(M) \text{ è vuoto}\}$

Linguaggio associato a un problema di decisione

Quali sono gli elementi del complemento $\overline{E_{TM}}$ di E_{TM}?

Linguaggio associato a un problema di decisione

Quali sono gli elementi del complemento $\overline{E_{TM}}$ di E_{TM}?

$\overline{E_{TM}} = \{\langle M \rangle \mid M \text{ è una MdT e } L(M) \neq \emptyset\} \cup \{y \mid y \text{ non è la codifica di una MdT}\}$.

Linguaggio associato a un problema di decisione

Studieremo i linguaggi associati ai problemi di decisione in relazione con la classe dei linguaggi decidibili e la classe dei linguaggi Turing riconoscibili.

Linguaggio associato a un problema di decisione

Studieremo i linguaggi associati ai problemi di decisione in relazione con la classe dei linguaggi decidibili e la classe dei linguaggi Turing riconoscibili.

La macchina di Turing **verifica**, come passo preliminare, che l'input corrisponde a una codifica dell'input del problema. Prosegue la computazione solo in questo caso.

Linguaggio associato a un problema di decisione

Studieremo i linguaggi associati ai problemi di decisione in relazione con la classe dei linguaggi decidibili e la classe dei linguaggi Turing riconoscibili.

La macchina di Turing **verifica**, come passo preliminare, che l'input corrisponde a una codifica dell'input del problema. Prosegue la computazione solo in questo caso.

Spesso nella descrizione della macchina di Turing, il passo preliminare corrispondente a questa verifica è omesso.

Problemi di decisione

Diremo che un problema di decisione è:

- **decidibile** se il linguaggio associato è decidibile

Problemi di decisione

Diremo che un problema di decisione è:

- **decidibile** se il linguaggio associato è decidibile
- **semidecidibile** se il linguaggio associato è Turing riconoscibile

Problemi di decisione

Diremo che un problema di decisione è:

- **decidibile** se il linguaggio associato è decidibile
- **semidecidibile** se il linguaggio associato è Turing riconoscibile
- **indecidibile** se il linguaggio associato non è decidibile.

Problemi di decisione

Il problema di decisione ispirato dal decimo problema di Hilbert

Input: *p polinomio a coefficienti interi.*
Domanda: *p ha una radice intera?*

Il problema di decisione ispirato dal decimo problema di Hilbert

Input: *p polinomio a coefficienti interi.*
Domanda: *p ha una radice intera?*

Il corrispondente linguaggio associato è

$$D = \{\langle p \rangle \mid p \text{ è un polinomio a coefficienti interi avente una radice intera}\}$$

Il problema di decisione ispirato dal decimo problema di Hilbert

Input: *p polinomio a coefficienti interi.*
Domanda: *p ha una radice intera?*

Il corrispondente linguaggio associato è

$$D = \{\langle p \rangle \mid p \text{ è un polinomio a coefficienti interi avente una radice intera}\}$$

Sappiamo che D non è decidibile e quindi il corrispondente problema di decisione non ammette una soluzione algoritmica, è indecidibile.

Motivazioni per lo studio di questi problemi:

Motivazioni per lo studio di questi problemi:

- Essere consapevoli che non tutti i problemi possono essere risolti mediante algoritmi/programmi.

Motivazioni per lo studio di questi problemi:

- Essere consapevoli che non tutti i problemi possono essere risolti mediante algoritmi/programmi.

I problemi indecidibili non sono esoterici o lontani dai problemi di interesse informatico. Esempi di problemi indecidibili sono:

- Stabilire se un programma si arresta.
- Stabilire se due programmi forniscono lo stesso output.
- Stabilire se un programma è un virus.

Risultati principali del Capitolo IV, sezione 4.2

- Metodo della diagonalizzazione.
  Permette di provare l'esistenza di linguaggi che non sono Turing-riconoscibili (prova di esistenza non costruttiva).

Risultati principali del Capitolo IV, sezione 4.2

- Metodo della diagonalizzazione.
  Permette di provare l'esistenza di linguaggi che non sono Turing-riconoscibili (prova di esistenza non costruttiva).
- Macchina di Turing universale. Anticipò alcuni sviluppi fondamentali in informatica:

Risultati principali del Capitolo IV, sezione 4.2

- Metodo della diagonalizzazione.
  Permette di provare l'esistenza di linguaggi che non sono Turing-riconoscibili (prova di esistenza non costruttiva).
- Macchina di Turing universale. Anticipò alcuni sviluppi fondamentali in informatica:
  - Compilatore C (risp. C^{++}, Java) scritto in C (risp. in C^{++}, in Java)

Risultati principali del Capitolo IV, sezione 4.2

- Metodo della diagonalizzazione.
  Permette di provare l'esistenza di linguaggi che non sono Turing-riconoscibili (prova di esistenza non costruttiva).
- Macchina di Turing universale. Anticipò alcuni sviluppi fondamentali in informatica:
  - Compilatore C (risp. C^{++}, Java) scritto in C (risp. in C^{++}, in Java)
  - Computer a programma memorizzato

Risultati principali del Capitolo IV, sezione 4.2

- Metodo della diagonalizzazione.
  Permette di provare l'esistenza di linguaggi che non sono Turing-riconoscibili (prova di esistenza non costruttiva).
- Macchina di Turing universale. Anticipò alcuni sviluppi fondamentali in informatica:
  - Compilatore C (risp. C^{++}, Java) scritto in C (risp. in C^{++}, in Java)
  - Computer a programma memorizzato
- Proveremo che un particolare linguaggio

  $$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM che accetta la parola } w\}$$

  non è decidibile.

Risultati principali del Capitolo IV, sezione 4.2

- Metodo della diagonalizzazione.
  Permette di provare l'esistenza di linguaggi che non sono Turing-riconoscibili (prova di esistenza non costruttiva).
- Macchina di Turing universale. Anticipò alcuni sviluppi fondamentali in informatica:
  - Compilatore C (risp. C^{++}, Java) scritto in C (risp. in C^{++}, in Java)
  - Computer a programma memorizzato
- Proveremo che un particolare linguaggio

  $$A_{TM} = \{\langle M, w\rangle \mid M \text{ è una TM che accetta la parola } w\}$$

  non è decidibile.
- Esibiremo un linguaggio che non è Turing-riconoscibile.