Note

In alcun modo questi appunti sono completi, ma possono tornare utili per un ripasso pre-esame.

HALTing problem

$$HALT = \{\langle y, x\rangle \in \Sigma^* \times \Sigma^* \mid y = code(M) \text{ e } M \text{ ferma su } x\}$$

Teorema

$HALT$ è riconoscibile ma non decidibile

Dimostrazione

1. $HALT$ è riconoscibile

Basta usare una UTM con input $\langle code(M), x\rangle$

2. $HALT$ è decidibile

Assumiamo che $HALT$ sia decidicibile,e chiamiamo M_H la TM che lo decide
Ovvero che (se $y = code(M)$ per qualche M) accetta se M ferma su x, altrimenti rigetta

Definiamo una nuova MT M' che simula M_H su input $\langle z, z\rangle$, con $z \in \Sigma^*$

- Se M_H rigetta, accetta
- Se M_H accetta, cicla

Proviamo ora ad eseguire M' su input $code(M')$

Notiamo che avremo una contraddizione sia se assumessimo che accetti o che rigetti

L'assunzione di avere M_H che decide è quindi impossibile, quindi $HALT$ non è decidibile

Teorema

Il complemento $HALT^-$ non è riconoscibile da nessuna TM

Teorema

Se $HALT^-$ fosse riconoscibile, allora $HALT$ sarebbe decidibile

Dimostrazione

Supponiamo per assurdo che $HALT^-$ sia riconoscibile, e costruiamo una TM M_{H-} la TM che lo riconosce

Chiamiamo M_{HR} la TM che riconosce $HALT$

Costruiamo ora M_H che decide $HALT$

- Simula ***in parallelo*** M_{HR} e M_{H-}
  - Se M_{HR} ferma, accetta
  - Se M_{H-} ferma, rigetta

Perciò M_H decide $HALT$

Sapendo che $HALT$ non è decidibile, abbiamo che $HALT^-$ non è riconoscibile

Teorema

Se L e L^- sono riconoscibili, allora L è decidibile

Dimostrazione

$L = HALT$

Corollario

I linguaggi riconoscibili non sono chiusi per complemento

Mapping Reduction

Definizione

L' è mapping-riducibile a L, scritto $L' \leq L$, se esiste una TM che computa la funzione (totale) $f : \Sigma^* \to \Sigma^*$ tale che $x \in L' \iff f(x) \in L$

Teorema

Se $L' \leq L$ e L è decidibile, allora L' è decidibile

Corollario

Se $L' \leq L$ e L' è indecidibile, allora L è indecidibile

Quindi, per dimostrare che L è indecidibile, basta mostrare che $HALT \leq L$

Corollario

Se L è decidibile e L' non lo è, allora $L' \not\leq L$

Teorema

Se $L' \leq L$ e L è riconoscibile, allora L' è riconoscibile

Corollario

Se $L' \leq L$ e L' non è riconoscibile, allora L non è riconoscibile

Corollario

Se L è riconoscibile e L' non lo è, allora $L' \not\leq L$

Empty Tape Halting problem (ETH)

$$ETH = \{x \in \Sigma^* \mid x = code(M) \text{ e } M \text{ ferma su } \epsilon\}$$

Teorema

ETH è indecidibile

Dimostrazione

Basta ridurre $HALT$ a ETH

Costruiamo una funzione f computabile che: $\langle y, x\rangle \in HALT \iff f(\langle y, x\rangle) \in ETH$

- $y \neq code(M) \quad \forall M$, allora $f(\langle y, x\rangle) = y \notin ETH$
- $y = code(M)$, allora $f(\langle y, x\rangle) = code(M_{M,x})$, con la TM costruita così:
  1. $M_{M,x}$ entra in loop su ogni stringa non vuota
  2. Su input ϵ, scrive x sul nastro e simula M su x

Full Language problem (FL)

$$FL = \{x \in \Sigma^* \mid x = code(M) \text{ e } M \text{ ferma su ogni input}\}$$

Teorema

FL è indecidibile

Dimostrazione

Riduciamo $HALT$ a FL

Si dimostra analogamente a ETH, ma $M_{M,x}$ è costruita così:

1. $M_{M,x}$ cancella il suo input
2. Scrive x sul nastro e simula M su x

Poichè se M ferma su x allora $M_{M,x}$ ferma su input arbitrario (viene cancellato), allora abbiamo $HALT \leq FL$

Equivalence Problem (EQ)

$$EQ = \{\langle y, x\rangle \in \Sigma^* \times \Sigma^* \mid x = code(M), y = code(M'), \text{ e } M, M' \text{ computano la stessa funzione parziale}\}$$

Teorema

EQ è indecidibile

Dimostrazione

Riduciamo FL a EQ. Usiamo FL perchè lavora su singole stringhe e può essere più comodo

La funzione parziale è definita così:

- Se $z \neq code(M) \quad \forall M$, allora $f(z) = \langle z, z\rangle \notin ETH$
- Se $z = code(M)$, allora $f(z) = \langle code(M_1), code(M_2)\rangle$, dove M_1 ed M_2 sono costruite così:
  - M_1 esegue M sul suo input e restituisce 1 se M si ferma, altrimenti va in loop
  - M_2 restituisce 1 per ogni input

Così definite se $z \in FL$ allora fermerà su ogni input, quindi M_1 restituirà sempre 1. Allora M_1 ed M_2 sono equivalenti

Teorema

EQ non è riconoscibile

Dimostrazione

Riduciamo $HALT^-$, che sappiamo essere non riconoscibile, a EQ, ovvero $HALT^- \leq EQ$

Per ridurre dobbiamo costruire f tale che $\langle y, x\rangle \in HALT^- \iff f(\langle y, x\rangle) \in EQ$

Ovvero $\langle y, x\rangle \in HALT \iff f(\langle y, x\rangle) \in EQ^-$

- $y \neq code(M) \quad \forall M$, allora prendiamo M' qualsiasi e $f(\langle y, x\rangle) = \langle code(M'), code(M')\rangle \in EQ$, quindi $f(\langle y, x\rangle) \notin EQ^-$
- $y = code(M)$, allora $f(\langle y, x\rangle) = \langle code(M_1), code(M_2)\rangle$, con:
  - M_1 esegue M su x e si ferma se M si ferma, altrimenti entra in un ciclo
  - M_2 cicla sempre

In questo modo se $\langle y, x \rangle \in HALT$ allora M_1 si fermerà, e quindi M_1 ed M_2 saranno diverse, quindi $\notin EQ$ e quindi $\in EQ^-$

Se invece $\langle y, x \rangle \notin HALT$ allora M_1 non si fermerà, e quindi M_1 ed M_2 saranno equivalenti, quindi $\in EQ$ e quindi $\notin EQ^-$

Teorema

$$L_1 \leq L_2 \iff L_1^- \leq L_2^-$$

Corollario

$$L_1^- \leq L_2 \iff L_1 \leq L_2^-$$

Corollario

EQ^- non è riconoscibile

Infatti $HALT \leq FL \leq EQ$, quindi $HALT^- \leq EQ^-$

Tiling Problem

Sulle slide

Teorema

Il Tiling Problem è non riconoscibile

Dimostrazione

Riduciamo ETH^-, non riconoscibile, al Tiling Problem

Mostriamo che ogni TM può essere trasformata in un sistema di tiling Avremo che un passo di computaizone equivale ad una riga del sistema di tiling

Proprietà dei linguaggi

Funzione da un insieme di TM a $\{0, 1\}$, tale che $L_M = L_{M'} \Rightarrow P(M) = P(M')$

Una proprietà è NON TRIVIALE se esiste una TM M tale che $P(M) = 1$ e una TM M' tale che $P(M) = 0$

Teorema di Rice

Se P è language property non triviale, allora il problema "M ha proprietà P" è indecidibile

Dimostrazione

Per contraddizione. Dimostriamo che se "M ha proprietà P" fosse decidibile, allora $HALT$ sarebbe decidibile

- Consideriamo P, e assumiamo che $P(M_) = 0$
- Poichè P è non triviale, possiamo considerare una TM M_P tale che $P(M_P) = 1$
- Fissiamo M e x come parametri e costruiamo la TM $M_{M,x}$ che prende in input z, simula M su x e se M ferma, simula M_P su z, e se M_P ferma allora accetta

In ogni altro caso la TM costruita ciclerà

Provando sia che M fermi su x sia che non, otteniamo che, se potessimo decidere se $M_{M,x}$ ha P, allora potremmo decidere il problema della fermata

Nota

Il Teorema di Rice riguarda proprietà di linguaggio, non proprietà algoritmiche.

Riguarda funzioni (specifiche), non programmi (implementazioni)

Cardinalità dei problemi irrisolvibili

Sulle slide

Complessità (tempo)

Sulle slide

Classe P

Classe di linguaggi decidibili da una TM (deterministica, ad un nastro) in tempo polinomiale.

Tesi di Church-Turing rafforzata

Ogni modello di calcolo deterministico fisicamente realizzabile può essere simulato da una TM (deterministica, con un nastro), con overhead al più polinomiale

Classe EXP

Linguaggi decidibili in tempo irragionevole, ovvero $TIME(2^{n^k})$

Classe NP

Classe di linguaggi decidibili da una TM (non deterministica, ad un nastro) in tempo polinomiale.

CLIQUE

$$CLIQUE = \{\langle G, k\rangle \mid \text{Il grafo } G \text{ contiene un clique di } k \text{ nodi}\}$$

Un CLIQUE, dato un grafo indiretto, è un sotto-grafo dove tutti i noti sono collegati tra loro da un arco

CLIQUE è in NP

1. Seleziona non deterministicamente un sottoinsieme C di k nodi di G
2. Verifica che G colleghi tutti i nodi di C tramite archi.
   - Se sì, accetta
   - Se no, rigetta

Poly Mapping Reduction

L' è poly (mapping-)riducibile a L, scritto $L' \leq_p L$, se esiste una TM che computa *in tempo polinomiale* la funzione (totale) $f : \Sigma^* \to \Sigma^*$ tale che $x \in L' \iff f(x) \in L$

In altre parole, $L' \leq_p L$ se $L' \leq L$ e la riduzione che lo testimonia è computabile in tempo polinomiale

Teorema

Se $L' \leq_p L$ e $L \in P$, allora $L' \in P$

Teorema

Per L non triviale in P, $L' \leq_p L$

SAT e 3SAT

Formula booleana

Formula costruita a partire da variabili, dalle loro negazioni e combinazioni di letterali, separate da $\vee$ o $\wedge$

La forma è soddisfacibile se esiste un assegnamento di valore alle sue variabili che le dia valore 1

Una formula è una clausola se è una disgiunzione ($\vee$) di variabili (positive o negate)

Una formula booleana è in forma normale congiunta (cnf) se è una congiunzione ($\wedge$) di clausole

Una formula booleana è 3cnf se è in cnf e ogni clausola contiene esattamente 3 letterali

SAT

$$SAT = \{\langle F\rangle \mid F \text{ è una formula booleana soddisfacibile}\}$$

3SAT

$$3SAT = \{\langle F\rangle \mid F \text{ è una formula booleana 3cnf soddisfacibile}\}$$

Teorema

$3SAT \leq_p CLIQUE$

Interessante perchè metter in relazione due problemi all'apparenza molto diversi tra loro (uno formule booleane e l'altro grafi)

Dimostrazione

$\langle F\rangle \in 3SAT \iff f(\langle F\rangle) \in CLIQUE$

L'idea è di tradurre le formule in grafi, mettendo ogni tripla dentro un 3-CLIQUE, in modo tale da "mimare" il comportamento di variabili e clausole
Un CLIQUE in $f(\langle F\rangle)$ corrsiponderà ad un assegnamento che soddisfa $\langle F\rangle$

- Nodi: G ha $3k$ nodi, suddivisi in triple. Ogni tripla corrisponde ad una clausola di F
- Archi: gli archi di G collegano tutte le coppie di nodi, eccetto:
  - Se fanno parte della stessa tripla
  - Se il nodo a cui vanno a collegarsi è la negazione di sè stessi, quindi una coppia $(x_1, \neg x_1)$ non può esserci

Dimostriamo che $\langle F\rangle \in 3SAT \quad \Rightarrow \quad f(\langle F\rangle) \in CLIQUE$

Se F è soddisfacibile, almeno un letterale per ogni clausola è vero. Se consideriamo i nodi corrispondenti avremo un CLIQUE di k nodi, dove k è il numero di triple, poichè saranno connessi tra loro per costruzione

Ora dimostriamo che $\langle F\rangle \in 3SAT \quad \Leftarrow \quad f(\langle F\rangle) \in CLIQUE$

Se $f(\langle F\rangle) = \langle G, k\rangle$ e G contiene un clique S di k nodi, definiamo un assegnamento A di valori di verità alle variabili di F, dando valore vero ai letterali nel CLIQUE

Per costruzione notiamo che:

- A non è una contraddizione, infatti nodi etichettati con un letterale e la sua negazione non sono collegati
- Ogni nodo di S appartiene ad una tripla diversa rispetto ad altri in S, perciò A assegna valore vero esattamente ad un letterale per clausola

Perciò A rende vera F

Notiamo che avere più $CLIQUE$ non ci da fastidio, infatti è esattamente avere più soluzioni alla formula di $3SAT$

NP-completezza

Un linguaggio L è NP-completo quando è in NP e $\forall L' \in NP,\ L' \leq_p L$

Teorema di Cook-Levin

$$SAT = \{\langle F \rangle \mid F \text{ è una formula booleana soddisfacibile}\}$$

SAT è NP-completo

Corollario: Se SAT è in P, allora $P = NP$

Dimostrazione

Dimostrare che SAT è in NP è immediato, presa la formuala possiamo costruire una TM che scelga (non deterministicamente) un assegnamento A, e accettare se A rende vera F

Dimostrare che ogni linguaggio in NP è poly-riducibile a SAT è più complicato. La dimostrazione completa è sulle slide

Teorema

$3SAT$ è NP-completo

Dimostrazione

Analoga a quella per SAT

Corollario

$CLIQUE$ è NP-completo

Dimostrazione

Abbiamo visto che $3SAT \leq_p CLIQUE$, e che $3SAT$ è NP-completo. Allora anche CLIQUE sarà NP-completo

Complessità di spazio

Sia M una TM che si ferma su tutti gli input. La sua complessità di spazio è definita come la funzione $t : \mathbb{N} \to \mathbb{N}$, dove $t(n)$ è il massimo numero di celle del nastro che M visita su arbitrari input di lunghezza n

Definiamo $SPACE(t(n))$ come la collezione di tutti i linguaggi decidibili da una TM (deterministica, ad un nastro) in spazio $O(t(n))$

Definiamo $NSPACE(t(n))$ come la collezione di tutti i linguaggi decidibili da una TM (non deterministica, ad un nastro) in spazio $O(t(n))$

$PSPACE$

Classe di linguaggi decidibili da una TM (deterministica, ad un nastro) in spazio polinomiale.

$NPSPACE$

Classe di linguaggi decidibili da una TM (non deterministica, ad un nastro) in spazio polinomiale.

Lemma

SAT è in $PSPACE$

Dimostrazione

Considera la TM M che prova ogni possibile assegnamento A e testa se F è vera.

Ogni iterazione può essere eseguita in spazio lineare al numero di variabili di F, e prima dell'iterazione successiva possiamo cancellare il nastro.

$NP \subseteq PSPACE$

Ovviamente vale che $P \subseteq PSPACE$

Ma vale anche che $NP \subseteq PSPACE$

Dimostrazione

Sia L in NP, per il teorema di Cook-Levin $L \leq_p SAT$, e SAT è in $PSPACE$.

Allora possiamo facilmente dimostrare che L sia in $PSPACE$

$PSPACE$-completo

Un linguaggio L è $PSPACE$-completo se è in $PSPACE$ e se $\forall L',\ L' \leq_p L$

Enunciato

Una formula è un enunciato se tutte le variabili sono vincolate da un quantificatore

TQBF

$$TQBF = \{\langle F \rangle \mid F \text{ è un enunciato booleano vero}\}$$

Teorema

$TQBF$ è $PSPACE$-completo

Dimostrazione

È facile dimostrare che $TQBF$ è in $PSPACE$ (come per SAT), dobbiamo poi dimostrare che qualsiasi L in $PSPACE$ è poly-riducibile a $TQBF$

Sulle slide

Teorema di Savitch

Per qualsiasi funzione $t : \mathbb{N} \to \mathbb{N}$, abbiamo:

$$NSPACE(t(n)) \subseteq SPACE(t(n^2))$$

Corollario

$$NSPACE = PSPACE$$

Non sorprende, visto che nelle TM non deterministiche non dobbiamo "ricordarci" nulla della computazione di un ramo precedente (se non la parte comune)

Gerarchia delle classi di complessità

Funzione costruibile

$f : \mathbb{N} \to \mathbb{N}$, dove $f(n) \geq O(\log n)$, è $SPACE$-costruibile se possiamo calcolare in spazio $O(f(n))$ la funzione

$$1^n \mapsto \langle f(n) \rangle$$

Dove $\langle f(n) \rangle$ è la codifica binaria di $f(n)$

Ad esempio n^2 è $SPACE$-costruibile (Prende in input 1^n, traduce in binario, ovvero $\langle n \rangle$, contando il numero di 1, e ritorna come output $\langle n^2 \rangle$, usando qualsiasi metodo di moltiplicazione con sè stesso)

Teorema della gerarchia di spazio

Per ogni $f : \mathbb{N} \to \mathbb{N}$ $SPACE$-costruibile, esiste un linguaggio L decidibile in spazio $O(f(n))$, ma non in spazio $o(f(n))$

Dimostrazione

L verrà definito da un algoritmo ALG, che usa spazio $O(f(n))$ ed è costruito in modo tale da assicurare che L è diverso da ogni linguaggio decidibile in spazio $o(f(n))$

Si sfutta la tecnica della diagonalizzazione. Su input $code(M)$, ALG simula M su $code(M)$ in una porzione di nastro $f(n)$, e accetta se e solo se M ferma e rigetta. Ovvero fa l'opposto di quello che farebbe M

Il resto sulle slide

Gerarchie di tempo

Sulle slide