Aniello Murano

Problemi decidibili e non decidibili

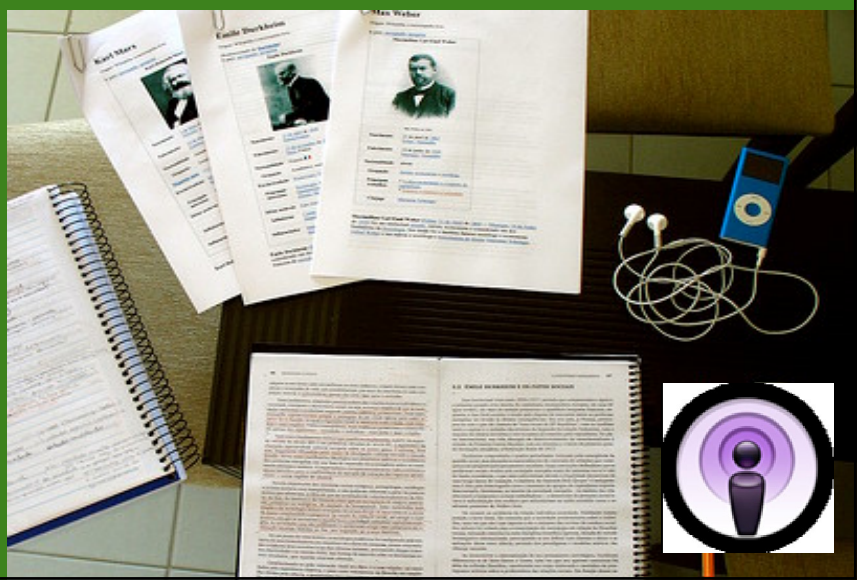
Lezione n.7
Parole chiave:
Decidibilità

Corso di Laurea:
Informatica

Codice:

Email Docente:
murano@ na.infn.it

A.A. 2008-2009



Overview

- In questa lezione mostreremo alcuni problemi decidibili per le macchine di Turing.
- In particolare, mostreremo su automi finiti che possono essere decisi da macchine di Turing (membership, vuoto, equivalenza, ecc)
- Mostreremo poi che già il semplice caso della membership (cioè stabilire se una data parola è accettata da una data macchina di Turing è non decidibile, sebbene Turing riconoscibile.
- Per mostrare l'indecidibilità del problema precedente utilizzeremo la tecnica della diagonalizzazione di Cantor.

Turing Machine e NFA

- Abbiamo già detto che le macchine di Turing sono modelli computazionali più potenti degli NFA.
- In particolare è possibile definire degli algoritmi per testare se
  - Un DFA/NFA accetta una data stringa
  - Un linguaggio di un DFA/NFA è vuoto o meno
  - Due automi finiti sono equivalenti (cioè accettano lo stesso linguaggio).
- Tutti questi problemi possono essere rifrasati in termini di linguaggi. Per esempio, il problema di accettazione per un DFA può essere espresso tramite un linguaggio A_{DFA} contenente la codifica di tutti i possibili DFA e le stringhe che essi accettano nel modo seguente:

$$A_{DFA} = \{ (B, w) \mid B \text{ è un DFA che accetta la stringa in input } w\}$$

- Il problema di verificare se UN DFA *B* accetta un input w equivale al problema di verificare se ***(B,*** *w) è un elemento di* A_{DFA}.
- Similmente, possiamo riformulare gli altri problemi in termini di verifica di appartenenza di un elemento in un linguaggio.
- Mostrare che il linguaggio A_{DFA} è decidibile equivale a mostrare che il relativo problema computazionale è decidibile.

A_{DFA} è decidibile

- Una macchina di Turing M in grado di decidere A_{DFA} è la seguente:
  - *M prende in input una codifica di B e una di w dove B* è un DFA e w è una stringa. Più precisamente, la codifica di B è la codifica dei suoi 5 componenti. Ogni codifica algoritmica (rigorosa) va bene!
  - M simula B su w, similmente a come visto per la macchina di Turing universale*. In pratica M legge B e w e in base alla transizione simulata aggiorna la configurazione corrente*
  - Se B finisce in uno stato di accettazione alla fine della lettura della stringa w (cioè la configurazione finale è di accettazione) allora M transisce nello stato di accettazione "yes"; altrimenti M rifiuta transendo nello stato "no".
- Un procedimento simile si può applicare nel caso di
  - NFA: l'input di M sarà la codifica del DFA corrispondente.
  - A_{REX} = { (R, w) | R è una espressione regolare che genera la stringa w}: l'input di M sarà la codifica del DFA corrispondente a R.

Il controllo del vuoto per i DFA è decidibile

- Sia E_{DFA} {A | A è un DFA e L(A) = Φ}
- **Teorema:** E_{DFA} è un linguaggio decidibile.
- **Prova:** Una macchina di Turing in grado di decidere questo linguaggio è quella mostrata nelle lezioni precedenti per il calcolo della raggiungibilità. In pratica, presa una codifica del DFA, la macchina di Turing verifica se seguendo le regole di transizione è possibile raggiungere uno stato finale da uno iniziale.

  L'algoritmo opera con un marcatore degli stati raggiungibili.

  Se ad un certo punto non è possibile marcare nuovi stati e non si è ancora raggiunto uno stato si accettazione per il DFA allora la macchina di Turing non accetta.
- La decidibilità del vuoto per un DFA permette di decidere anche l'equivalenza di due DFA. Infatti, ricordando che i DFA sono chiusi rispetto al complemento, all'unione e all'intersezione, l'equivalenza di due DFA A e B equivale a decidere il vuoto di

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

Problemi decidibili per i PDA e i DPDA

- Alcuni dei problemi visti per gli NFA sono decidibili anche per i PDA, ma richiedono prove più complesse.
- Per esempio, per verificare il vuoto di un PDA non si può semplicemente simulare le sue configurazioni visto che sono infinite.
- Di contro però, le tavole di transizione dei PDA sono finite (tutti gli oggetti che li compongono sono finiti). Se si opera dunque sui componenti, allora si può verificare se una configurazione finale è raggiungibile o meno da una iniziale.
- Infatti, sono decidibili per i PDA (e dunque anche per i DPDA) il problema dell'appartenenza (di una stringa al linguaggio accettato da un NPDA o generato da una (D)CFL) e del vuoto.
- Per testare l'equivalenza di due PDA non possiamo semplicemente usare la stessa idea mostrata per i DFA visto che i PDA non sono chiusi rispetto al complemento. Per i PDA il problema dell'equivalenza è indecidibile.
- Per i DPDA il problema dell'equivalenza*,** è invece decidibile. Anche per i DPDA non è possibile usare lo stesso ragionamento usato per i DFA visto che i DPDA non sono chiusi rispetto a unione e intersezione.

Note bibliografiche:

*Geraud Senizergues. The equivalence problem for deterministic pushdown automata is decidable. In Automata, languages and programming (Bologna, 1997), volume 1256 of Lecture Notes in Comput. Sci., pp. 671- 681. Springer, Berlin, 1997.

**Geraud Senizergues. L(A) = L(B)? A simplied decidability proof. Theoret. Comput. Sci., 281(1-2):555-608, 2002. Selected papers in honour of Maurice Nivat.

Problemi indecidibili per le macchine di Turing

- **Teorema:** Il problema dell'appartenenza è indecidibile per la macchina di Turing.
- In seguito mostriamo la prova della sua indecidibilità così come di altri problemi. Questo al fine di introdurre tecniche di prova di indecidibilità.
- Il problema di determinare se una data macchina di Turing accetta una data stringa si può ridurre al problema di decisione del linguaggio seguente A_{TM}, in linea con quanto fatto per gli NFA:

$$A_{TM} = \{(M, w) \mid M \text{ è una macchina di Turing e } M \text{ accetta } w\}$$

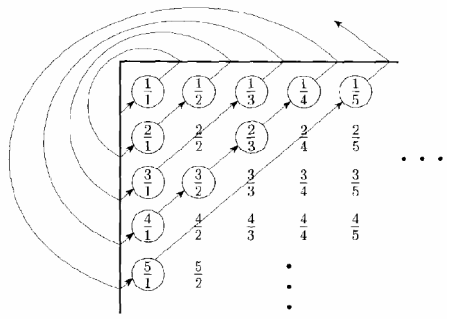
- Prima di mostrare che A_{TM} è indecidibile, mostriamo che è Turing-riconoscibile (Turing-recognizable).
- Questo mostra formalmente che i riconoscitori sono più potenti dei decisori.
- Una macchina di Turing U capace di accettare A_{TM} è la seguente:
- Su un input (M, w), dove M è la codifica di una TM e w è una stringa:
  1. U simula M su w.
  2. Se M non entra mai in uno stato di accettazione, allora U accetta; se M non entra mai in uno stato di rifiuto, allora U accetta.
- Si noti che U cicla su un input (M, w) se M cicla su w, il che dimostra perché U non decide A_{TM}.
- Se U avesse un modo per determinare che M non si fermerà mai su un certo input w, allora U potrebbe non accettare un tale input.
- Come vedremo in seguito, A_{TM} permette di dimostrare che il **problema della fermata** (**halting problem**) è indecidibile. Mostreremo cioè che nessuna macchina di Turing è in grado di determinare se un'altra macchina di Turing si fermerà prima o poi, o meno.

Il metodo di Cantor

- Per provare la non decidibilità di A_{TM}, si può ricorrere al metodo della diagonalizzazione introdotto da Georg Cantor nel 1873.
- A quell'epoca, Cantor si occupava della misurazione degli insiemi.
- Confrontare due insiemi finiti è semplice: basta contare gli elementi che essi contengono.
- Nel caso di insiemi infiniti, invece, non possiamo utilizzare questo metodo, perché chiaramente non finirebbe mai.
- Cantor propose un metodo alternativo basato sull'osservazione che due insiemi (finiti o infiniti) hanno la stessa taglia se gli elementi di un insieme possono essere accoppiati con gli elementi dell'altro.
- Prima di descrivere formalmente questo metodo, introduciamo alcune notazioni matematiche sulle funzioni:
- Si assuma di avere due insiemi A e B, e una funzione f da A a B. La funzione f è detta corrispondenza (biettiva) se
  - $f(a) \neq f(b)$ se e solo se $a \neq b$
  - per ogni $b \in B$ c'è un $a \in A$ tale che $f(a)=b$.
- Due insiemi A and B sono della stessa taglia se c'è una corrispondenza $f:A \rightarrow B$.
- In una corrispondenza, ogni elemento di A corrisponde ad un solo elemento di B e ogni elemento di B ha un unico elemento di A che corrisponde ad esso.

Insiemi contabili

- Con il metodo di Cantor, è possibile mostrare, ad esempio, che l'insieme dei numeri naturali e l'insieme dei numeri naturali pari hanno la stesa taglia.
- Infatti la corrispondenza f tra i due insiemi è semplicemente *f(n)* = 2n.
- **Definizione:** Un insieme è contabile se è finito oppure se ha la stessa taglia dell'insieme dei numeri naturali N.
- Si consideri l'insieme dei numeri razionali positivi Q = {m/n | m,n ∈ N}. Questo insieme è contabile?
- La risposta è **sì** perché esiste un modo per associare tutti gli elementi di Q agli elementi di N tramite una corrispondenza. Il metodo è il seguente:
- Si dispongono tutti i numeri di Q in una matrice infinita dove la i-esima riga contiene tutti i numeri con numeratore i e la j-esima colonna tutti i numeri con denominatore j.
- La matrice si può trasformare in una lista leggendo i suoi elementi in diagonale. Si noti che non leggiamo gli elementi per riga perché le righe sono infinite, altrimenti partendo dalla prima riga non visiteremmo mai la seconda.



La corrispondenza di N e Q

Insiemi non contabili

- Non tutti gli insiemi infiniti possono essere messi in corrispondenza con N. Questi insiemi sono detti non contabili.
- Per esempio, l'insieme dei numeri reali R non è contabile. La prova procede per assurdo, mostrando che esiste un termine che non ha corrispondenza in N.
  - Il numero si costruisce nel seguente modo: da tutti i numeri f(i), si costruisce il termine t= $0,c_1c_2$.... dove ogni c_i è tale da essere differente dalla i-esima cifra decimale di f(i). Per esempio se f(1)= 4,14... e f(2)=6,45... allora t può essere 0,27... Questa costruzione assicura che t non corrisponde a nessun f(i).
- Trasportando questo risultato nella teoria della computazione, se uno mostra che **i linguaggi non sono contabili**, mentre **le macchini di Turing sono contabili**, segue che alcuni linguaggi non sono decidibili o addirittura Turing-riconoscibile.
- Per mostrare che l'insieme delle macchine di Turing è contabile si osservi che l'insieme di tutte le stringhe **Σ* è contabile**. La funzione di corrispondenza è tra le stringhe di lunghezza i e l'insieme dei numeri necessari per listare tutte le parole di quella lunghezza.
- L'insieme delle macchine di Turing è contabile perchè ogni macchina è una codifica su Σ.

I linguaggi sono non contabili

- Per mostrare che l'insieme dei linguaggi è non contabile, prima si osservi che l'insieme B delle sequenze infinite binarie è non contabile (prova simile al caso dei reali).
- Sia C l'insieme di tutti i linguaggi sull'alfabeto Σ. Mostriamo adesso che anche C è non contabile, dando una corrispondenza con B.
- Sia $\Sigma^* = \{s_1, s_2, s_3, \ldots\}$. Adesso associamo i linguaggi di C alle stringhe di B in modo che ciascun linguaggio A in C abbia un'unica sequenza in B.
- La sequenza binaria è così costituita: l'i-esimo bit della sequenza è 1 se $s_i \in A$ e 0 altrimenti (formalmente questa sequenza è chiamata la X_A caratteristica di A). Per esempio, se A è il linguaggio delle stringhe che iniziano per 0 sull'alfabeto $\Sigma = \{0,1\}$, la sua caratteristica X_A è la seguente:

$$\begin{aligned} \Sigma^* &= \{\ \varepsilon\ ,\ 0\ ,\ 1\ ,\ 00\ ,\ 01\ ,\ 10\ ,\ 11\ ,\ 000\ ,\ 001\ ,\ \cdots\ \}\ ; \\ A &= \{\ \quad 0\ ,\ \quad 00\ ,\ 01\ ,\ \quad\quad 000\ ,\ 001\ ,\ \cdots\ \}\ ; \\ \chi_A &= \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad \cdots \quad . \end{aligned}$$

- Esercizio per gli studenti: Provare che X_A è una biezione

Prova della indecidibilità di A_{TM}

- Supponiamo per assurdo che il problema dell'appartenenza è decidibile. Dimostreremo che tale ipotesi produce una contraddizione.
- Se A_{TM} è decidibile, per la universalità esiste una macchina di Turing H che decide M, cioè, data M (più precisamente, una sua codifica) e un qualsiasi input w di M, la macchina H si ferma e accetta se M accetta w, mentre H si ferma e non accetta se M non riesce ad accettare w. Schematicamente H(M,w) si comporta nel modo seguente:
  - accetta se la computazione di *M* con ingresso *w* termina con accettazione,
  - rifiuta se la computazione di *M* con ingresso *w* non accetta (cicla o rifiuta).
- Adesso, si consideri una nuova macchina di Turing D che utilizzi H come subroutine, tale che D chiama H per stabilire come si comporta M su input M e restituisce l'opposto del valore ottenuto. In pratica, D si comporta come segue:
  - rifiuta se la computazione di *M* con ingresso *M* termina con accettazione,
  - accetta se la computazione di *M* con ingresso *M* non accetta.
- **Nota:** Il fatto di eseguire una macchina sulla propria descrizione non deve preoccupare. Questo è simile al caso in cui un programma lavora con se stesso come input. Per esempio, un compilatore è un programma che trasforma altri programmi ed entrambi possono essere scritti con lo stesso linguaggio.

Prova del teorema della fermata

- Cosa succede se a D diamo in input la stessa macchina D (in pratica la sua codifica).
- Ci chiediamo cioè se D accetta o meno con input D?
- Abbiamo che *D(D) accetta se D* non accetta *D. Viceversa, rifiuta D quando D accetta D.*
- Questa è ovviamene una contraddizione. Dunque, H e D non possono esistere
- In pratica, abbiamo dimostrato, in base alla definizione di D, che D con input D da origine a un calcolo che termina se e soltanto se il calcolo di D per l'input D non termina, il che è palesemente assurdo.
- Ne consegue quindi che una macchina che si comporta come H non può esistere, e che quindi, se è vera la tesi di Church, non può esistere un algoritmo che permette di decidere il problema dell'appartenenza.
- Dunque il problema dell'appartenenza è indecidibile (o non ricorsivo).

L'halting problem e la tecnica della diagonalizzazione (1)

- Per meglio rendersi conto di questo risultato, consideriamo il metodo della diagonalizzazione di Cantor.
- Ancora una volta, si assuma che H sia in grado di decidere A_{TM}, che D, costruita a partire da H, su input M accetta esattamente quando M non accetta l'input M. Infine, si dia in input a D lo stesso D. Allora si ha che
  - H accetta *(M, w) esattamente quando M accetta w*
  - *D rifiuta M esattamente quando M accetta M (come input)*
  - *D rifiuta D quando D accetta D (contraddizione)*
- Vediamo adesso come la diagonalizzazione porta a mostrare un assurdo, sotto l'ipotedi che A_{TM} è decidibile
- Si consideri la tabella dei comportamenti di *H* e *D. In questa tabella riportiamo sulle righe le macchine di Turing M_1, M_2 ecc, e sulle colonne le loro descrizioni*
- Ogni combinazione riga i colonna j dice se una macchina M_i accetta o meno M_j. In particolare in (i,j) scriviamo *accept se* M_i accetta o meno M_j e lasciamo la cella vuota se M_i rifiuta o cicla su quell'input.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

- ,

L'halting problem e la tecnica della diagonalizzazione (2)

- Nella figura in alto al lato, ogni cella (i, j) rappresenta il risultato di H sugli input della figura mostrata nella slide precedente. Dunque, se M_3 no accetta M_2, nella cella (3,2) scriviamo reject, perché H rifiuta l'input (M_3, M_2).
- Nella figura in basso al lato, aggiungiamo la riga e la colonna D alla figura in alto.
- Si noti che D computa l'opposto dei valori presenti sulla diagonale. La contraddizione arriva in corrispondenza del punto interrogavo dove il valore deve essere l'opposto di se stesso.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*	
M_4	*accept*	*accept*	*reject*	*reject*	
$\vdots$			$\vdots$		

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	*reject*		*accept*	
$\vdots$			$\vdots$		$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		?	
$\vdots$			$\vdots$				$\ddots$

Un linguaggio non Turing-recognizable

- Completiamo questa lezione con due importanti risultati.
  1. Mostriamo una condizione necessaria e sufficiente per un linguaggio affinché sia ricorsivo(recursive)
  2. Mostriamo l'esistenza di linguaggi che non sono Turing-riconoscibile (Turing-recognizable).
- Definizione: Un linguaggio è co-Turing-riconoscibile (co-Turing-recognizable) se e solo se il suo complemento è Turing-riconoscibile (Turing-recognizable).
- **Teorema:** Un linguaggio è decidibile se e solo se è **Turing-riconoscibile** e **co-Turing-riconoscibile**.
- **Prova:** Ci sono due direzioni da provare. Dapprima assumiamo che A sia decidibile. Questa direzione segue dal fatto che il complemento di un linguaggio decidibile è decidibile e dal fatto che ogni linguaggio decidibile è anche Turing-riconoscibile.
- Per l'altra direzione, se A e il suo complemento comp(A) sono Turing-riconoscibile, possiamo usare una Macchina di Turing a 2 nastri dove sul primo decidiamo A e sul secondo decidiamo comp(A). Data una qualsiasi parola w essa appartiene ad A o a comp(A). Qualsiasi sia il caso, si ha che comunque la macchina su w si fermerà, per definizione di accettazione.
- **Corollario:** Comp(A_{TM}) non è Turing-riconoscibile.
- **Prova:** Noi sappiamo che A_{TM} è Turing-riconoscibile. Se anche comp(A_{TM}) fosse Turing-riconoscibile, allora per il teorema precedente anche A_{TM} sarebbe decidibile. Ma noi sappiamo che A_{TM} non è decidibile. Dunque, comp(A_{TM}) non può essere Turing-riconoscibile.

Gerarchia dei linguaggi