Aniello Murano

Problemi non decidibili e riducibilità

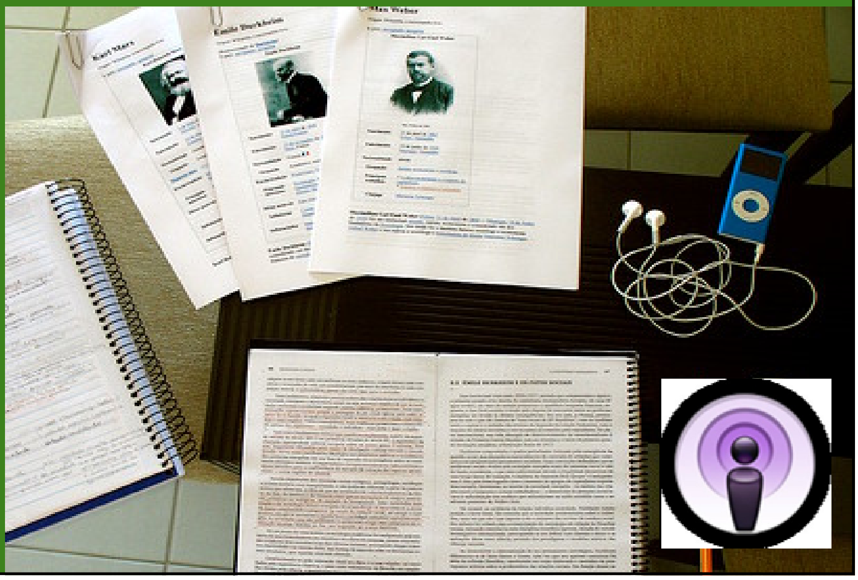
Lezione n.8
Parole chiave:
Riduzione

Corso di Laurea:
Informatica

Codice:

Email Docente:
murano@ na.infn.it

A.A. 2008-2009



Overview

- Nelle lezioni precedenti, abbiamo mostrato alcuni problemi decidibili ed altri indecidibili per le macchine di Turing.
- Tra quelli indecidibili, abbiamo mostrato A_{TM} che rappresenta il problema della membership per le macchine di Turing
- In questa lezione mostreremo un metodo fondamentale per provare che l'indecidibilità dei problemi: La **riducibilità**.
- Con questo metodo, proveremo l'indecidibilità dei seguenti problemi per le macchine di Turing:
- Il vuoto
- L'equivalenza
- La regolarità

Riduzione

- La riduzione è un modo per convertire un problema A in un altro problema B (A si riduce a B) in modo tale che una soluzione per B può essere usata per risolvere A
- Quando A (problema noto) è riducibile a *B (nuovo problema),* risolvere A non può essere più difficile della risoluzione di B perché una soluzione di B da una soluzione anche ad A. ($A \leq B$)
- In termini della teoria della calcolabilità, se A è riducibile a B e B è decidibile, anche A è decidibile.
- In modo equivalente, se A è indecidibile ed è riducibile a B, B è indecidibile.
- Quest'ultima versione è la chiave per dimostrare che molti problemi sono indecidibili.
- In breve, il metodo che useremo per dimostrare che un problema A è indecidibile sarà quello di mostrare che altri problemi già noti essere indecidibili sono riducibili ad A.
- Notare che la riducibilità dice che non conosciamo una soluzione di A o B considerandoli singolarmente, ma sappiamo solo che una soluzione per A si ottiene da una soluzione di B.
- La riducibilità gioca un ruolo fondamentale nella classificazione dei problemi attraverso la decidibilità e conseguentemente nella teoria della complessità.

Riduzione di A_{TM} a $HALT_{TM}$

- Abbiamo già stabilito l'indecidibilità di A_{TM}, il problema di determinare se dato un input una Macchina di Turing accetta.
- Consideriamo allora un problema simile, $HALT_{TM}$, il problema di determinare se su un certo input la Macchina di Turing si ferma (con accept o reject). Per dimostrare l'indecidibilità di $HALT_{TM}$ possiamo sfruttare l'indecidibilità di A_{TM} riducendo A_{TM} a $HALT_{TM}$.

- La definizione di $HALT_{TM}$ è la seguente:

$HALT_{TM}$ = {(M, w) | M è una TM e M si ferma su input w}.

Dimostrazione di indecidibilità per $HALT_{TM}$

- Dimostriamo quindi che $HALT_{TM}$ è indecidibile.
- Assumiamo che $HALT_{TM}$ è decidibile e usiamo questa assunzione per dimostrare che A_{TM} è decidibile, contraddicendo quello che abbiamo detto nella lezione precedente.
- L'idea chiave è mostrare che A_{TM} è riducibile ad $HALT_{TM}$.
- Supponiamo di avere una TM R che decide $HALT_{TM}$.
- Usiamo R per costruire S, una TM che decide A_{TM} nel seguente modo:
- S prende in input una codifica di una TM M e una stringa w:
  1. Facciamo girare la TM R su input (M, w).
  2. Se R rifiuta (cioè M va in reject o su w non si ferma), allora S rifiuta (perché (M,w) non è presente nel linguaggio di A_{TM}).
  3. Se R accetta (cioè M su w o accetta o rifiuta), allora si simula M su w finchè non si ferma.
  4. Se M accetta, allora R accetta; se M rifiuta, allora R rifiuta.
- Chiaramente, se R decide $HALT_{TM}$, allora S decide A_{TM}. Poichè A_{TM} è indecidibile, allora anche $HALT_{TM}$ deve essere indecidibile.

Decidere il vuoto per TM è indecidibile

- Nel resto di questa lezione mostriamo altre riduzione per provare l'indecidibilità di vari linguaggi.
- Consideriamo $E_{TM} = \{ (M) \mid M \text{ è una TM e } L(M) = \Phi \}$.
- Mostriamo con la riduzione che E_{TM} è indecidibile.
- Assumiamo per ottenere una contraddizione che E_{TM} è decidibile e in seguito mostriamo che A_{TM} è decidibile. Otteniamo cosi una contraddizione.

Decidere il vuoto per TM è indecidibile(2)

- Consideriamo una TM R che decide E_{TM}. Usiamo R per costruire una TM S che decide A_{TM}.
- Come si comporterà S quando riceve in input *(M, w)?*
- Un'idea per S è quella di far girare R con input (M) e vedere se accetta.
- Se accetta, significa che sappiamo che L(M) è vuoto e di conseguenza M non accetta w. Ma se R rifiuta (M), allora sappiamo che L(M) non è vuoto e che M accetta qualche stringa, ma ancora non sappiamo se M accetta una particolare stringa w.
- Abbiamo dunque bisogno di un'idea differente.
- Invece di far girare R su <M>, facciamo girare R su una modifica di <M>. Modifichiamo M in modo che garantisca che M rifiuta tutte le stringhe tranne w, tenendo presente che anche su input w la macchina lavora sempre allo stesso modo.
- A questo punto usiamo R per determinare se la macchina modificata riconosce il linguaggio vuoto.
- Ora l'unica stringa che la macchina accetta è w, in questo modo il suo linguaggio sarà non vuoto se e solo se accetta w.
- Se R accetta quando riceve in input una descrizione della macchina modificata, allora sappiamo che la macchina modificata non accetta nulla e che M non accetta w.

DIMOSTRAZIONE

- Dimostriamo la correttezza della costruzione fatta nella slide precedente. Chiamiamo M1 la macchina appena costruita, che opera nel modo seguente:

  M1= "Su input x:
  1. Se x ≠ w, rifiuta.
  2. Se x = w, si considera M su input w e accetta se M accetta;

- M1 verifica se x = w in modo ovvio: scandisce l'input e lo confronta carattere per carattere con la stringa w per determinare se sono uguali.

- Assumiamo che la TM R decide E_{TM}. Si costruisce allora la TM S che decide A_{TM} come descritto di seguito:

  S=" Su input *(M,w)*, dove M è una codifica di una TM M e w una stringa:
  1. Utilizza la descrizione di M e w per costruire la TM M1 appena descritta.
  2. Fa girare R su input (M1).
  3. Se R accetta, allora rifiuta; Se R rifiuta, allora accetta; "

- S deve essere in grado di computare una descrizione di M1 da una descrizione di M e w. Per fare questo basta aggiungere degli stati in più ad M per verificare che x=w.
- Se R è un decisore per E_{TM}, allora S è un decisore per A_{TM}. Un decisore per A_{TM} non può esistere, così sappiamo che E_{TM} è indecidibile.

L'Equivalenza di due TM è indecidibile

- Usando ragionamenti simili a quelli fatti nelle diapositive precedenti, è possibile dimostrare l'indecidibilità di vari linguaggi.
- Per esempio, si consideri il seguente:

- **EQ_{TM} = { (M_1,M_2) | M_1 e M_2 sono TM e $L(M_1)=L(M_2)$}.**

- Usando il concetto di riduzione dal problema per E_{TM}, si dimostra facilmente che EQ_{TM} è indecidibile.
- *R* decisore per EQ_{TM}
- *S* decisore per E_{TM}
- *S* = "su ingresso <*M*>:
  - costruisce M_1 come la MdT t.c. $L(M1) = \varnothing$
  - esegui *R* su ingresso <*M*, M_1>
  - se *R* accetta → S accetta (cioè $L(M) = L(M1) = \varnothing$) altrimenti → rifiuta ($L(M) \neq L(M1)$ quindi il linguaggio riconosciuto da *M* non è vuoto)."
- Dato che *S* non può esistere (E_{TM} non decidibile) non può esistere neanche *R*, quindi EQ_{TM} non è decidibile (dimostrazione per contraddizione).

Regular è indecidibile

- Si consideri il seguente linguaggio

$REGULAR_{TM}$ = {(M)| *M* è una TM and *L(M)* è un linguaggio regolare}

- Anche in questo caso, usando il concetto di riduzione, si può mostrare che $REGULAR_{TM}$ è indecidibile.

- Assumiamo per ottenere una contraddizione che $REGULAR_{TM}$ è decidibile e in seguito mostriamo che A_{TM} è decidibile. Otteniamo cosi una contraddizione.

- Per ogni coppia (*M,w*) *costruiamo una MdT M′ che si comporta nel modo seguente su ogni x:*
  1. Se *x* è *della forma* (0^n1^n), allora M′ accetta.
  2. altrimenti, simula *M su w*.
  3. Se *M* accetta *w*, allora M′ accetta *x*.
  4. Se M rifiuta, allora M′ rifiuta *x*.
- $L(M') = \Sigma^*$ *se M* accetta *w*.
- $L(M') = (0^n1^n)$ se *M* non accetta *w*.
- Quindi, $L(M')$ è regolare se e solo se *M* accetta *w*.

- Provare che i seguenti linguaggi sono indecidibili:
  - L={(M) | M è una TM che accetta "000"}
  - CONTEXT-FREE$_{TM}$ = {(M) | M è una TM e L(M) è context-free}
  - NOTREGULAR$_{TM}$ = {(M) | M è una TM e L(M) non è regolare}