Aniello Murano

Mapping reducibility

Lezione n.10

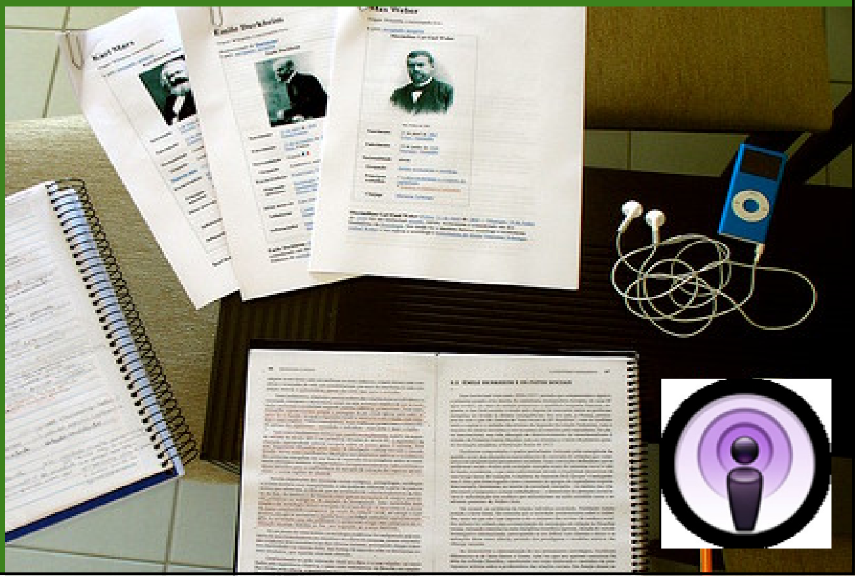
Parole chiave:
Mapping reducibility

Corso di Laurea:
Informatica

Codice:

Email Docente:
murano@ na.infn.it

A.A. 2008-2009



Mapping Reducibility

- Ridurre un problema A ad un problema *B* usando una **mapping reducibility** ⇒ esiste una funzione calcolabile che converte istanze del problema A in istanze del problema B.
- Se abbiamo una tale funzione di conversione, chiamata riduzione, possiamo risolvere A risolvendo B.
- Una qualsiasi istanza del problema A può essere risolta usando prima la riduzione per convertire A in una istanza di B e poi risolvendo B.

Funzioni calcolabili

- Definizione:

  una funzione $f : \Sigma^* \rightarrow \Sigma^*$ è una funzione calcolabile (***computable function)*** se esiste qualche macchina di Turing *M che, su ogni input w*, si ferma con $f(w)$ sul suo nastro.

- Esempio:

  Tutte le consuete operazioni aritmetiche sugli interi sono funzioni calcolabili. Per esempio,possiamo costruire una macchina che prende in input $\langle m, n \rangle$ e restituisce $m+n$, la somma di m e n.

Definizione formale di mapping reducibility

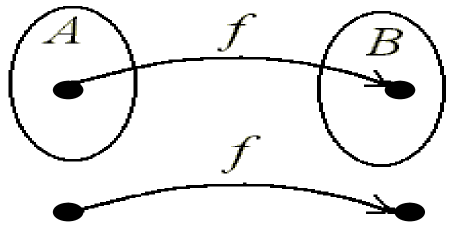
- Definizione:

  Un linguaggio *A* è ***mapping reducible*** ad un linguaggio *B*, si scrive $A \leq_m B$, se esiste una funzione calcolabile $f : \Sigma^* \rightarrow \Sigma^*$, tale che per ogni w,

$$w \in A \Leftrightarrow f(w) \in B.$$

- La funzione f è chiamata una riduzione di A a B.

La seguente figura illustra una mapping reducibility



La Funzione *f* riduce *A* a *B*.

1. Una mapping reduction da *A* a *B* fornisce un modo per convertire il problema di testare l'appartenenza ad A nel problema di testare l'appartenenza a B.
2. Per testare se $w \in A$, usiamo la riduzione *f* per mappare *w* con $f(w)$ e testiamo se $f(w) \in B$.

Decidibilità e $\leq_m$

- Teorema:

  Se $A \leq_m B$ e B è decidibile, allora A è decidibile.
- *Dimostrazione*:

  Sia M un decisore per B e f una riduzione da A a B. Un decisore N per A è il seguente:

  N = "Su input w:

  1. Calcola $f(w)$.
  2. esegui M su input $f(w)$ e restituisci l'output di M."

  Chiaramente, se $w \in A$, allora $f(w) \in B$ perchè f è una riduzione da A a B. Dunque, M accetta $f(w)$ se $w \in A$.
- Corollario:

  If $A \leq_m B$ e A è indecidibile, allora B è indecidibile.
- Dimostrazione:

  se A è indecidibile e B è decidibile allora si contraddice il precedente teorema

Mapping reducibility e complementazione

- Teorema:
  Se $A \leq_m B$, allora anche per i complementi $(\Sigma^* \setminus A) \leq_m (\Sigma^* \setminus B)$

- Dimostrazione:
  sia f la funzione di riduzione da A a B con $w \in A \Leftrightarrow f(w) \in B$.
- Per questa stessa funzione vale che $w \in (\Sigma^* \setminus A) \Leftrightarrow f(w) \in (\Sigma^* \setminus B)$,
  per ogni $w \in \Sigma^*$

Mapping reducibility per $HALT_{TM}$

- Rivediamo alcune delle dimostrazioni che usano il metodo della riduzione per fare alcuni esempi di mapping riducibilità
- *Cominciamo con l'indecidibilità di $HALT_{TM}$* attraverso la riduzione da A_{TM}
- Possiamo dimostrare una mapping riduzione usando una funzione calcolabile *f che prende un input di forma (M, w)* e ritorna un output di forma *(M', w'),* dove

$$\langle M, w\rangle \in A_{TM} \text{ se e solo se } \langle M', w'\rangle \in HALT_{TM}.$$

- La seguente macchina *F* calcola una riduzione *f*.

  F = "su input $\langle M, w\rangle$:

  1. Si costruisce la seguente macchina *M′*, tale che:

     M′ = "Su input *x*:

     1. si esegue *M* su *x*.
     2. se *M* accetta, allora M′ *accetta*.
     3. se *M* rifiuta, allora M′ entra in loop."

  2. Restituisci $\langle M', w\rangle$."

Mapping reducibility per EQ_{TM}

- Consideriamo ora l'indecidibilità di EQ_{TM} attraverso una riduzione da E_{TM} una mapping riduzione f da E_{TM} a EQ_{TM} mappa gli input $\langle M \rangle$ con gli output $\langle M, M_1 \rangle$, dove M_1 è la macchina che va in reject su tutti gli input.

$$\langle M \rangle \in E_{TM} \text{ se e solo se } \langle M, M_1 \rangle \in EQ_{TM}.$$

- La seguente macchina F calcola una riduzione f.

  F = "su input $\langle M \rangle$:

  1. Costruisce M_1 che rifiuta tutti gli input.
  2. Costruisce M′ tale che

     M' = "su input (M, M_1) :

     1. se M accetta, *accetta*. (perché $L(M)=L(M_1)$)
     2. se M rifiuta, rifiuta."
  3. Restituisce $\langle M', M_1 \rangle$."

Riconoscibilità and $\leq_m$

- Teorema

  se $A \leq m$ B e B è Turing-riconoscibile, allora A è Turing-riconoscibile.
- Dimostrazione:

  sia M la TM che riconosce B e f la funzione di riduzione da A a B. Allora M su input w:

  1. Calcola f(w)
  2. Simula M su f(w) e restituisce lo stesso risultato.

- Dalla definizione di f: $w \in A$ è equivalente con $f(w) \in B$.
- M "accetta" f(w) se $w \in A$, e
- M "rifiuta" f(w)/non si ferma su f(w) se $w \notin A$.

- Corollario:

  se $A \leq_m B$ e A non è Turing-riconoscibile, allora B non è Turing-riconoscibile.
- dimostrazione:

  Il linguaggio A non è Turing-riconoscibile e se B fosse riconoscibile si contraddirebbe il teorema precedente

Riconoscibilità e $\leq_m$

- Teorema :
  se $A \leq_m B$ e A non è co-Turing riconoscibile, allora B non è co-Turing-riconoscibile.
- Dimostrazione:
  Se A non è co-Turing-riconoscibile, allora il complemento $(\Sigma^* \backslash A)$
  non è Turing-riconoscibile
  da $A \leq_m B$ sappiamo che $(\Sigma^* \backslash A) \leq_m (\Sigma^* \backslash B)$.
- Dal corollario precedente: $(\Sigma^* \backslash B)$ non è Turing-riconoscibile, quindi B non è co-Turing-riconoscibile.

EQ_{TM}

- Teorema:
  EQ_{TM} non è né Turing-riconoscibile né co-Turing-riconoscibile.
  Dimostrazione:
  Come prima cosa mostriamo che EQ_{TM} non è Turing-riconoscibile, per fare questo mostriamo che $A_{TM} \leq_m comp(EQ_{TM})$.
  (questo equivale a dimostrare che $comp(A_{TM}) \leq_m EQ_{TM}$ e noi sappiamo che $comp(A_{TM})$ è co-Turing-riconoscibile)
  La funzione di riduzione f è la seguente:
  F = "su input $\langle M, w \rangle$ dove M è una TM e w una stringa:
  1. Costruisce le seguenti 2 macchine M_1 e M_2.
     M_1 = "su ogni input:
     1. *Rifiuta*."
     M_2 = "su ogni input:
     1. esegui M su w. se M accetta, *accetta*."
  2. ritorna $\langle M_1, M_2 \rangle$."

comp(EQ_{TM}) non è Turing-riconoscibile

Per mostrare che comp(EQ_{TM}) non è Turing-riconoscibile vediamo che $A_{TM} \leq_m EQ_{TM}$.

La seguente G calcola la funzione di riduzione g:

G = "su input $\langle M, w \rangle$ dove M è una TM e w una stringa:

1. Costruisce le seguenti 2 macchine M_1 e M_2.

   M_1 = "su ogni input:

   1. *Accetta*."

   M_2 = "su ogni input:

   1. esegui M su w. se M accetta , *accetta*."

2. ritorna $\langle M_1, M_2 \rangle$."

Linguaggi senza mapping riduzione

- L'indecidibilità di E_{TM} mostrata nella precedente lezione, mostra invece la differenza tra la nozione formale di "mapping reducibility" e quella informale di riducibilità mostrata precedentemente
- La dimostrazione mostra che E_{TM} è indecidibile tramite una riduzione da A_{TM}.
- Vediamo se riusciamo a trasformare questa riduzione in una mapping riduzione.
- Dall'originale riduzione possiamo facilmente costruire una funzione f che prende in input *(M, w)* e restituisce *(M_1)*, dove M_1 è la TM tale che L(M_1) è non vuoto (L(M_1) = {w}) iff w ∈ L(M).
- Dunque *f* è una "mapping riduzione" da A_{TM} a comp(E_{TM}).
- Questo dimostra ancora che E_{TM} è indecidibile perché la decidibilità non è affetta dalla complementazione, ma f non è certamente una mapping reduction da A_{TM} a E_{TM}.
- Infatti è possibile mostrare che una tale riduzione non può esistere:
- Supponiamo per assurdo che $A_{TM} <_m E_{TM}$ tramite una riduzione *f. Per definizione di mapping* reducibility segue che comp(A_{TM})$<_m$*comp(*E_{TM}) attraverso la stessa funzione di riduzione f.
- Ma comp(E_{TM}) è Turing-riconoscibile e comp(A_{TM}) non è Turing-riconoscibile, contraddicendo il precedente teorema sulla turing-riconoscibilità nella slide 9.