

Informatica Teorica



Anno Accademico 2022/2023

Fabio Zanasi

<https://www.unibo.it/sitoweb/fabio.zanasi>

Settima lezione

Nelle puntate precedenti

Abbiamo dimostrato che:

1. Il problema della fermata (*HALT*) è indecidibile.
2. Il complemento del problema della fermata (*HALT*⁻) non è riconoscibile.
3. (2) può essere dimostrato come conseguenza di (1).

In questa lezione

Introduciamo una tecnica semplice e generale per dimostrare l'indecidibilità/non-riconoscibilità di un problema:
mapping-reduction.

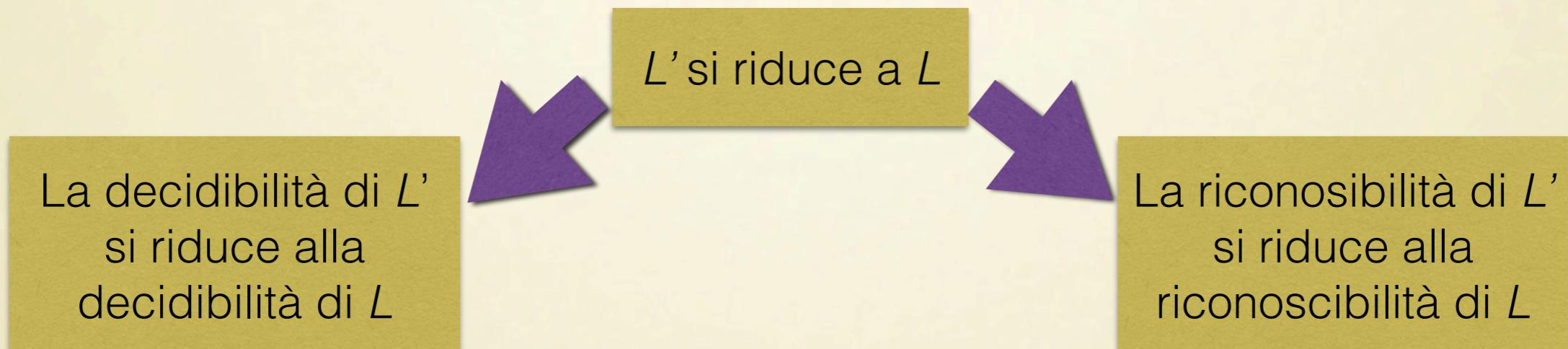
Mapping-reduction: perché

La mapping-reduction consente di **ridurre** le istanze di un problema a quelle di un altro problema.

Tale tecnica ci rivela che, nel confrontare due problemi la **decidibilità/riconoscibilità non è essenziale nel processo di prova**. Eventualmente, può essere derivata come una conseguenza.

Mapping-reduction: perché

Schematicamente:



Rispetto ad un tentativo diretto di investigare la calcolabilità di L o L' , questo approccio è maggiormente strutturato: per prima cosa, ci concentriamo sulla relazione i problemi.

Mapping-reduction

Definizione. Siano L e L' linguaggi sull'alfabeto Σ , diciamo che L' è mapping-riducibile a L , scritto $L' \leq L$, se esiste una TM che computa la funzione (totale) $f : \Sigma^* \rightarrow \Sigma^*$ tale che

$$x \in L' \Leftrightarrow f(x) \in L$$

Sostanzialmente, f converte il problema di appartenenza per L' nel problema di appartenenza per L .

Lettura intuitiva di $L' \leq L$: L è difficile almeno quanto L' .

Riduzione e Decidibilità

La *mapping-reducibility* non parla di decidibilità.
È però uno strumento efficace per mostrare che un linguaggio è (in)decidibile.

Teorema 1. Se $L' \leq L$ e L è decidibile, allora L' è decidibile.

Esercizio

Corollario 1. Se $L' \leq L$ e L' è indecidibile, allora L è indecidibile.

Quindi, per dimostrare che L è indecidibile, è sufficiente mostrare che $\text{HALT} \leq L$.

Corollario 2. Se L è decidibile e L' non lo è, allora $L' \not\leq L$.

Mapping-reduction in azione

Il problema della fermata sul nastro vuoto

Il problema della fermata su nastro vuoto (*ETH*) è definito dal linguaggio:

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

Teorema. *ETH* è indecidibile.

Piano della dimostrazione. È sufficiente ridurre *HALT* a *ETH*, così che l'indecidibilità di *HALT* implichì quella di *ETH*, via Corollario 1.

Il problema della fermata sul nastro vuoto

Dimostrazione. Costruiamo una funzione computabile f tale che:

f è computabile da
una TM

$$\langle y, x \rangle \in HALT \Leftrightarrow f(\langle y, x \rangle) \in ETH$$

La definizione di f è come segue. Su argomento $\langle y, x \rangle$:

- se $y \neq \text{code}(\mathcal{M})$ per ogni \mathcal{M} , allora $f(\langle y, x \rangle) = y \notin ETH$.
- se $y = \text{code}(\mathcal{M})$, allora $f(\langle y, x \rangle) = \text{code}(\mathcal{M}_{\mathcal{M}, x})$, dove $\mathcal{M}_{\mathcal{M}, x}$ è costruita come segue:
 1. $\mathcal{M}_{\mathcal{M}, x}$ entra in loop su ogni stringa non vuota.
 2. su input ε , scrive x sul nastro e simula \mathcal{M} su x .

$$\langle y, x \rangle \in HALT$$

\Leftrightarrow

$y = \text{code}(\mathcal{M})$ e \mathcal{M}
ferma su x .

\Leftrightarrow

$\mathcal{M}_{\mathcal{M}, x}$ ferma
su ε .

\Leftrightarrow

$f(\langle y, x \rangle) =$
 $\text{code}(\mathcal{M}_{\mathcal{M}, x})$ e
 $f(\langle y, x \rangle) \in ETH$

Il “full language” problem

Il “full language” problem (FL) è definito dal linguaggio seguente:

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

Teorema. FL è indecidibile.

Schema della Dimostrazione. Riduciamo HALT a FL .

Il “full language” problem

Dimostrazione. Costruiamo una funzione f computabile tale che:

$$\langle y, x \rangle \in HALT \Leftrightarrow f(\langle y, x \rangle) \in FL$$

Definiamo f come segue. Su argomento $\langle y, x \rangle$:

- se $y \neq \text{code}(\mathcal{M})$ per ogni \mathcal{M} , allora $f(\langle y, x \rangle) = y \notin FL$.
- se $y = \text{code}(\mathcal{M})$, allora $f(\langle y, x \rangle) = \text{code}(\mathcal{M}_{\mathcal{M}, x})$, dove $\mathcal{M}_{\mathcal{M}, x}$ è costruita come segue:
 1. $\mathcal{M}_{\mathcal{M}, x}$ cancella il suo input.
 2. scrive x sul nastro e simula \mathcal{M} su x .

$$\langle y, x \rangle \in HALT$$

 \Leftrightarrow

$$y = \text{code}(\mathcal{M}) \text{ e } \mathcal{M} \text{ ferma su } x.$$

 \Leftrightarrow

$$\mathcal{M}_{\mathcal{M}, x} \text{ ferma su un input arbitrario}$$

 \Leftrightarrow

$$f(\langle y, x \rangle) = \text{code}(\mathcal{M}_{\mathcal{M}, x}) \text{ e } f(\langle y, x \rangle) \in FL$$

L'equivalence problem

L'equivalence problem (EQ) per le TM è definito dal seguente linguaggio:

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

Teorema. EQ è indecidibile.

Piano della Dimostrazione. Riduciamo FL a EQ .

Osserva che avremmo potuto usare la riduzione da $HALT$ invece che da FL . Abbiamo cambiato per mostrare la flessibilità di questo approccio.

L'equivalence problem

Prova. E' sufficiente costruire una funzione computabile f tale che:

$$z \in FL \Leftrightarrow f(z) \in EQ$$

La definizione di f è come segue. Su argomento z :

- se $z \neq \text{code}(\mathcal{M})$ per ogni \mathcal{M} , allora $f(z) = \langle z, z \rangle \notin EQ$.
- se $z = \text{code}(\mathcal{M})$, allora $f(z) = \langle \text{code}(\mathcal{M}_1), \text{code}(\mathcal{M}_2) \rangle$, dove \mathcal{M}_1 e \mathcal{M}_2 sono definite come segue:
 - \mathcal{M}_1 esegue \mathcal{M} sul suo input e restituisce 1 se \mathcal{M} si ferma, e va in loop altrimenti.
 - \mathcal{M}_2 restituisce 1 per ogni input.

L'equivalence problem

Reminder

\mathcal{M}_1 esegue \mathcal{M} sul suo input e restituisce 1 se \mathcal{M} si ferma, altrimenti entra in loop.
 \mathcal{M}_2 restituisce 1 per ogni input.

$z \in FL$

\mathcal{M}_1 ferma su ogni
input con output 1.

$f(z) = \langle \text{code}(\mathcal{M}_1), \text{code}(\mathcal{M}_2) \rangle$
e $f(z) \in EQ$.

$z = \text{code}(\mathcal{M})$ e \mathcal{M} ferma
su ogni input.

\mathcal{M}_1 e \mathcal{M}_2 sono
equivalenti.



Riduzione e Riconoscibilità

Riduzione e Riconoscibilità

Ciò che è vero per riduzione e decidibilità vale anche per riduzione e riconoscibilità.

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 $L' \not\leq L$.

L'equivalence problem non è riconoscibile

Ricorda che l'*equivalence problem* (*EQ*) per TMs è:

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

Abbiamo dimostrato *EQ* indecidibile. Ora mostriamo che *EQ* non è riconoscibile.

Teorema. *EQ* non è riconoscibile.

Piano della Dimostrazione. Riduciamo *HALT*⁻ (che sappiamo essere un problema non-riconoscibile) a *EQ*.

L'equivalence problem non è riconoscibile

Dimostrazione. Ricorda la definizione di HALT^- :

$$\text{HALT}^- = \{\langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y \neq \text{code}(\mathcal{M}) \text{ per ogni } \mathcal{M} \text{ o} \\ y = \text{code}(\mathcal{M}) \text{ e } \mathcal{M} \text{ non} \\ \text{ferma su } x\}$$

Per la riduzione, abbiamo la funzione f

$$\langle y, x \rangle \in \text{HALT}^- \Leftrightarrow f(\langle y, x \rangle) \in EQ$$

equivalente a

$$\langle y, x \rangle \in \text{HALT} \Leftrightarrow f(\langle y, x \rangle) \in EQ^-$$

L'equivalence problem non è riconoscibile

Dimostrazione. Su $\langle y, x \rangle$ definiamo f come segue:

- se $y \neq \text{code}(\mathcal{M})$ per ogni \mathcal{M} , allora prendiamo \mathcal{M}' qualsiasi e consideriamo $f(\langle y, x \rangle) = \langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle$. Allora $f(\langle y, x \rangle) \in EQ$ e quindi $f(\langle y, x \rangle) \notin EQ^-$.
- altrimenti $y = \text{code}(\mathcal{M})$ e consideriamo $f(\langle y, x \rangle) = \langle \text{code}(\mathcal{M}_1), \text{code}(\mathcal{M}_2) \rangle$, dove \mathcal{M}_1 e \mathcal{M}_2 sono definite come:
 - \mathcal{M}_1 esegue \mathcal{M} su x e si ferma se \mathcal{M} si ferma, altrimenti entra in un ciclo.
 - \mathcal{M}_2 entra in un ciclo su ogni input.

L'equivalence problem non è riconoscibile

Reminder

\mathcal{M}_1 per ogni input esegue \mathcal{M} su x e si ferma se \mathcal{M} si ferma, altrimenti cicla.

\mathcal{M}_2 entra in un ciclo su ogni input.

$\langle y, x \rangle \in$
 $HALT$

\mathcal{M}_1 accetta una
stringa.

$f(\langle y, x \rangle) = \langle \text{code}(\mathcal{M}_1), \text{code}(\mathcal{M}_2) \rangle$
e $f(\langle y, x \rangle) \in EQ^-$

$y = \text{code}(\mathcal{M})$ e \mathcal{M} si ferma
su x .

\mathcal{M}_1 e \mathcal{M}_2 non
sono equivalenti.

Il linguaggio delle stringhe
accettate da \mathcal{M}_1 è o Σ^* o \emptyset , in
base al fatto che \mathcal{M} si fermi su x .

Ulteriori proprietà della mapping-reduction

Riduzione e Decidibilità

Teorema 1. Se $L' \leq L$ e L è decidibile, allora L' è decidibile.

Corollario 1. Se $L' \leq L$ e L' è indecidibile, allora L è indecidibile.

Corollario 2. Se L è decidable e L' no, allora $L' \not\leq L$.

Se L' è decidibile, la riduzione a qualche L è di scarsa utilità.

Teorema 2. Se L è un qualunque linguaggio non-triviale ($L \neq \Sigma^*$ e $L \neq \emptyset$), allora $L' \leq L$ per ogni L' decidibile.

Riduzione e Decidibilità

Teorema 2. Se L è un qualunque linguaggio non-triviale ($L \neq \Sigma^*$ e $L \neq \emptyset$), allora $L' \leq L$ per ogni L' decidibile.

Dimostrazione.

Poiché L è non-triviale, possiamo prendere $x \in L$ e $y \notin L$. Definiamo la funzione $f : \Sigma^* \rightarrow \Sigma^*$ come segue:

$$f(z) = x \text{ se } z \in L'$$

$$f(z) = y \text{ altrimenti, i.e. se } z \notin L'$$

Poiché L' è decidibile, f può essere implementata tramite una TM che simula internamente la TM per L' e restituisce x o y coerentemente. Inoltre, per definizione di f :

$$z \in L' \Leftrightarrow f(z) \in L$$

Riduzione come Relazione

La riduzione è una relazione tra linguaggi.

È riflessiva: per ogni L , $L \leq L$.

È transitiva: $L' \leq L$ e $L \leq L''$ implicano $L' \leq L''$.

Non è però simmetrica (sarebbe simmetrica se $L' \leq L$ implica $L \leq L'$). 

Il problema della fermata costituisce un conto esempio. Per T.2, per ogni L' decidibile, $L' \leq HALT$. Se anche $HALT \leq L'$, allora $HALT$ sarebbe decidibile per T.1. Contraddizione.

Quindi, la riduzione *non* è una relazione di equivalenza.

Riduzione e Complemento

Teorema. $L_1 \leq L_2$ se e solo se $L_1^- \leq L_2^-$

Esercizio

Corollario. $L_1^- \leq L_2$ se e solo se $L_1 \leq L_2^-$

Usato implicitamente nell'ultima prova con $L_1 = \text{HALT}$ e $L_2 = \text{EQ}$.

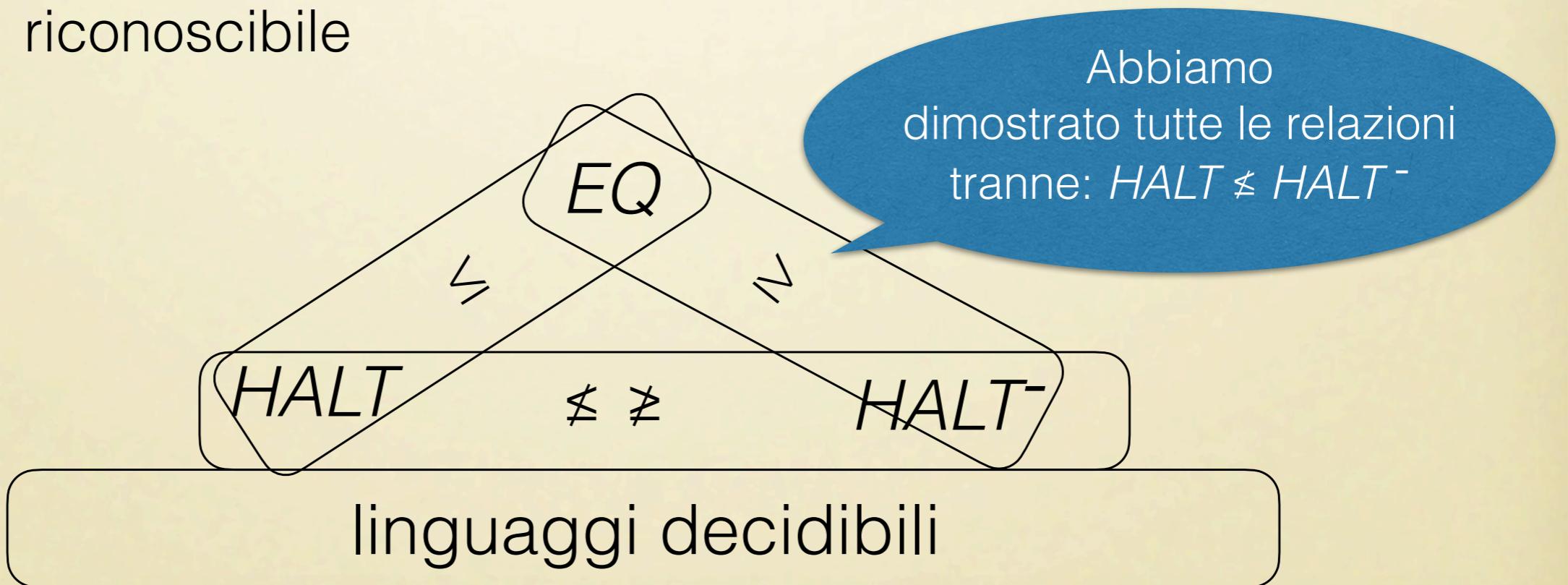
Mettendo insieme questi risultati, abbiamo senza ulteriore lavoro una prova che EQ^- non è riconoscibile.

Infatti, $\text{HALT} \leq \text{FL} \leq \text{EQ}$, quindi $\text{HALT}^- \leq \text{EQ}^-$ per teorema.

La non-riconoscibilità di HALT^- implica la non-riconoscibilità di EQ^- .

Gerarchia dei problemi

...	Decidibile
$HALT$	Indecidibile, ma riconoscibile
$HALT^-$	Non riconoscibile, ma il cui complemento è riconoscibile
EQ	Non riconoscibile, e non il cui complemento non è riconoscibile



Altre nozioni di riducibilità

Turing-riducibilità

La *mapping-reducibility* è solo uno dei modi in cui definire il concetto di problema riducibile a un altro problema.

Un altro è la **Turing-riducibilità**.

Un **oracolo** per un linguaggio L è uno strumento esterno (“black box”) capace di rispondere alla domanda “ $x \in L ?$ ” per ogni stringa x .

Un linguaggio L' è **Turing-riducibile** a L se, dato un oracolo per L , possiamo decidere L' .

Comparazione tra nozioni di riducibilità

La *mapping-reducibility* implica la Turing-riducibilità ma non vice-versa.

Esempio. Sappiamo che $\text{HALT}^- \not\leq \text{HALT}$, cioè HALT^- non è mapping-reducible a HALT .

Ma HALT^- è Turing-riducibile ad HALT . Infatti, se abbiamo un oracolo per la domanda “ $\langle y, x \rangle \in \text{HALT}?$ ”, questo può essere usato per decidere se $\langle y, x \rangle \in \text{HALT}^-$.