Riducibilità e Classe P

Tutorato 10: Limiti della Computabilità e Complessità

Automi e Linguaggi Formali

Gabriel Rovesti
Corso di Laurea in Informatica - Università degli Studi di Padova

23 Maggio 2025

Contents

1 Riducibilità **2**
1.1 Concetto Generale . . . 2
1.2 Proprietà Fondamentali . . . 2
1.3 Esempi Classici di Riduzione . . . 2

2 La Classe P **3**
2.1 Definizione . . . 3
2.2 Motivazioni Teoriche . . . 3
2.3 Esempi Canonici . . . 3
2.4 Tecnica Dimostrativa Tipica . . . 4
2.5 Relazione con le Riduzioni . . . 4

1 Riducibilità

1.1 Concetto Generale

Definizione

Una *riduzione* tra due problemi decisionali A e B è un procedimento effettivo che, data un'istanza w di A, produce un'istanza $f(w)$ di B tale che

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

Se la funzione f è calcolabile da una macchina di Turing che termina su tutti gli input, si parla di **riducibilità mediante funzione** e si scrive

$$A \leq_m B.$$

Concetto chiave

Riducendo un problema sconosciuto A ad uno già noto B è possibile trasferire proprietà di (in)decidibilità:

- Se $A \leq_m B$ e B è decidibile, allora A è decidibile.
- Se $A \leq_m B$ e A è indecidibile, allora B è indecidibile.

1.2 Proprietà Fondamentali

Teorema

Sia $A \leq_m B$.

a) Se B appartiene ad una classe di complessità chiusa verso le riduzioni (es. decidibile, Turing-riconoscibile), allora A è nella stessa classe.

b) Se A è *più difficile* (indecidibile, non Turing-riconoscibile, ecc.), allora lo è anche B.

Suggerimento

In pratica, per dimostrare che un nuovo problema B è indecidibile si riduce un problema classico come A_{TM} o $HALT_{TM}$ a B.

1.3 Esempi Classici di Riduzione

Dal problema dell'accettazione a quello della fermata $A_{TM} \leq_m HALT_{TM}$.
Data $\langle M, w\rangle$, costruiamo M' che su un input qualunque simula M su w e si ferma accettando se e solo se M accetta w.[1]

Dal problema del vuoto a quello dell'equivalenza $E_{TM} \leq_m EQ_{TM}$.

[1] Cfr. Bresolin, parte 16.

Per un dato $\langle M \rangle$ produciamo la coppia $\langle M_1, M \rangle$ dove M_1 rifiuta sempre: $L(M) = \emptyset \iff L(M_1) = L(M)$.

Errore comune

Confondere $A \leq_m B$ con $B \leq_m A$: la direzione della riduzione è essenziale.

2 La Classe P

2.1 Definizione

Definizione

La classe **P** è l'insieme dei linguaggi decidibili in tempo polinomiale da una TM deterministica a nastro singolo:

$$P = \bigcup_{k \geq 1} TIME(n^k).$$

Concetto chiave

Il polinomio rappresenta un limite "ragionevole" per il tempo di esecuzione: gli algoritmi in P sono considerati efficientemente computabili sui modelli reali di calcolo.[a]

[a] Bresolin, parte 18.

2.2 Motivazioni Teoriche

- Le differenze tra modelli deterministici *ragionevoli* sono al più polinomiali.[2]
- Un aumento esponenziale di tempo è invece sintomo di algoritmi a forza bruta o di problemi intrinsecamente difficili.

2.3 Esempi Canonici

- **PATH** $= \{\langle G, s, t \rangle \mid$ esiste un cammino da s a t in $G\}$: ricerca in ampiezza in $O(|V| + |E|)$.
- **RELPRIME** $= \{\langle x, y \rangle \mid \gcd(x, y) = 1\}$: algoritmo di Euclide in $O(\log \max\{x, y\})$.
- Ogni linguaggio context-free è in P tramite parsing CYK in $O(n^3)$.

[2] Bresolin, parte 17.

2.4 Tecnica Dimostrativa Tipica

Procedimento di dimostrazione

Per provare che un problema appartiene a P:

1. Descrivere un algoritmo in passi chiari.
2. Dimostrare che il numero di passi è $O(n^k)$ per qualche k.
3. Assicurarsi che ogni passo sia implementabile in tempo polinomiale su un modello deterministico standard.

2.5 Relazione con le Riduzioni

Se $A \leq_m B$ e $B \in P$, allora $A \in P$. Le riduzioni polinomiali sono fondamentali per definire le classi NP-completo, ma questo va oltre lo scopo di questo riassunto.

Errore comune

Ritenere che "polinomiale" equivalga sempre a "veloce": algoritmi con complessità $O(n^{10})$ possono essere impraticabili, ma dal punto di vista teorico rientrano comunque in P.