Indecidibili e NP-Hard - Approccio formale Bresolin

🎯 PANORAMICA GENERALE

Principio Fondamentale

Una **riduzione** $A \leq B$ significa: "Se posso risolvere B, allora posso risolvere A"

Corollario: Se A è difficile e $A \leq B$, allora B è almeno altrettanto difficile di A.

Due Tipi di Riduzione

1. **Riduzione mediante funzione** ($A \leq_m B$): per problemi **indecidibili**
2. **Riduzione polinomiale** ($A \leq_p B$): per problemi **NP-hard**

📚 PARTE I: RIDUZIONI PER INDECIDIBILITÀ

🔧 DEFINIZIONE FORMALE

Riduzione mediante funzione: $A \leq_m B$ se esiste funzione **calcolabile** $f : \Sigma \to \Sigma$ tale che:

```
∀w ∈ Σ*: w ∈ A ⟺ f(w) ∈ B
```

Proprietà: Se $A \leq_m B$ e A indecidibile, allora B è indecidibile.

🎯 METODOLOGIA STEP-BY-STEP

STEP 1: Setup della Riduzione

```
OBIETTIVO: Dimostrare che B è indecidibile
STRATEGIA: Mostrare A ≤ₘ B dove A è noto indecidibile

TEMPLATE:
"Dimostriamo che B è indecidibile mostrando che A ≤ₘ B"
```

STEP 2: Scelta del Problema Sorgente

Problemi base indecidibili (in ordine di preferenza):

1. **ATM** = {⟨M,w⟩ | M accetta w} - **IL PIÙ USATO**
2. **HALTTM** = {⟨M,w⟩ | M si ferma su w}
3. **ETM** = {⟨M⟩ | L(M) = ∅}
4. **REGULARTM** = {⟨M⟩ | L(M) è regolare}

STEP 3: Costruzione della Funzione f

```
TEMPLATE STANDARD:
F = "Su input ⟨parametri_A⟩:
1. [ESTRAZIONE] Estrai componenti dall'input di A
2. [COSTRUZIONE TM] Costruisci TM ausiliaria M':
   M' = "Su input x:
   [LOGICA CHE COLLEGA A E B]"
3. [OUTPUT] Restituisci ⟨M', altri_parametri⟩"
```

STEP 4: Dimostrazione di Correttezza

Schema bidirezionale obbligatorio:

```
CORRETTEZZA:
• CALCOLABILITÀ: F è chiaramente calcolabile [breve argomento]

• DIREZIONE (⇒): Se input ∈ A, allora f(input) ∈ B
  DIMOSTRAZIONE: [ragionamento caso per caso]

• DIREZIONE (⇐): Se f(input) ∈ B, allora input ∈ A
  DIMOSTRAZIONE: [ragionamento caso per caso]
```

STEP 5: Conclusione

```
CONCLUSIONE: Poiché A ≤ₘ B e A è indecidibile, allora B è indecidibile. □
```

🧩 PATTERN RICORRENTI

🔹 Pattern "Stringa Specifica"

Uso: Per problemi del tipo "TM accetta stringa s"

```
ESEMPIO: A₁₀₁₀ = {⟨M⟩ | 1010 ∈ L(M)}

F = "Su input ⟨M,w⟩:
   Costruisci M':
```

```
   M' = "Su input x:
   1. Se x ≠ 1010, RIFIUTA
   2. Se x = 1010:
      • Esegui M su w
      • Se M accetta w, ACCETTA
      • Se M rifiuta w, RIFIUTA"
   Restituisci ⟨M'⟩"

LOGICA: M accetta w ⟺ M' accetta 1010 ⟺ 1010 ∈ L(M')
```

◆ Pattern "Linguaggio Vuoto"

Uso: Per ETM e problemi correlati

```
ESEMPIO: ETM = {⟨M⟩ | L(M) = ∅}

F = "Su input ⟨M,w⟩:
   Costruisci M':
   M' = "Su input x:
   1. Se x ≠ w, RIFIUTA
   2. Se x = w:
      • Esegui M su w
      • Se M accetta w, ACCETTA
      • Se M rifiuta w, RIFIUTA"
   Restituisci ⟨M'⟩"

LOGICA: M accetta w ⟺ L(M') = {w} ≠ ∅ ⟺ ⟨M'⟩ ∉ ETM
```

◆ Pattern "Linguaggio Universale"

Uso: Per ALLTM = {⟨M⟩ | L(M) = Σ*}

```
F = "Su input ⟨M,w⟩:
   Costruisci M':
   M' = "Su input x:
   1. Esegui M su w
   2. Se M accetta w, ACCETTA x
   3. Se M rifiuta w, RIFIUTA x"
   Restituisci ⟨M'⟩"

LOGICA: M accetta w ⟺ L(M') = Σ* ⟺ ⟨M'⟩ ∈ ALLTM
```

🔹 Pattern "Proprietà Strutturale"

Uso: Per REGULARTM, CFTM, etc.

```
ESEMPIO: REGULARTM = {⟨M⟩ | L(M) è regolare}

F = "Su input ⟨M,w⟩:
   Costruisci M':
   M' = "Su input x:
   1. Se x ∈ 0*1*, ACCETTA
   2. Se x = 0ⁿ1ⁿ2ⁿ per n ≥ 1:
      • Esegui M su w
      • Se M accetta w, ACCETTA
   3. Altrimenti, RIFIUTA"
   Restituisci ⟨M'⟩"

LOGICA:
• M accetta w ⇒ L(M') = 0*1* ∪ {0ⁿ1ⁿ2ⁿ | n≥1} (non regolare)
• M non accetta w ⇒ L(M') = 0*1* (regolare)
```

📚 PARTE II: RIDUZIONI POLINOMIALI (NP-HARD)

🔧 DEFINIZIONE FORMALE

Riduzione polinomiale: $A \leq_p B$ se esiste funzione f **calcolabile in tempo polinomiale** tale that:

```
∀w ∈ Σ*: w ∈ A ⟺ f(w) ∈ B
```

Proprietà: Se $A \leq_p B$ e A è NP-hard, allora B è NP-hard.

🎯 METODOLOGIA STEP-BY-STEP

STEP 1: Setup della Riduzione

```
OBIETTIVO: Dimostrare che B è NP-hard
STRATEGIA: Mostrare A ≤ₚ B dove A è noto NP-hard
```

```
TEMPLATE:
"Dimostriamo che B è NP-hard mostrando che A ≤ₚ B"
```

STEP 2: Scelta del Problema Sorgente

Gerarchia problemi NP-hard (dal più usato):

1. **3SAT - IL PIÙ VERSATILE**
2. **IndependentSet** - per problemi su grafi
3. **VertexCover** - per problemi di copertura
4. **Clique** - per problemi su sottografi completi
5. **HamiltonianCycle** - per problemi di ordinamento

STEP 3: Costruzione della Riduzione

```
TEMPLATE PER 3SAT:
F = "Su input ϕ (formula 3CNF):
1. [ANALISI] Analizza struttura di ϕ (variabili, clausole)
2. [COSTRUZIONE] Costruisci istanza di B:
   • Per ogni variabile xᵢ: [gadget per variabile]
   • Per ogni clausola Cⱼ: [gadget per clausola]
   • [Connessioni tra gadget]
3. [PARAMETRI] Calcola parametri target
4. [OUTPUT] Restituisci istanza di B"
```

STEP 4: Dimostrazione di Correttezza

Schema bidirezionale con costruzioni esplicite:

```
CORRETTEZZA:
• POLINOMIALITÀ: f è calcolabile in tempo O(...) [analisi dettagliata]

• DIREZIONE (⇒): Se ϕ soddisfacibile, allora f(ϕ) ∈ B
  COSTRUZIONE: [come costruire soluzione per B da assegnamento di ϕ]

• DIREZIONE (⇐): Se f(ϕ) ∈ B, allora ϕ soddisfacibile
  COSTRUZIONE: [come estrarre assegnamento per ϕ da soluzione di B]
```

🧩 RIDUZIONI STANDARD

🔹 3SAT → IndependentSet

```
COSTRUZIONE GADGET:
• VERTICI: 3 vertici per clausola (uno per letterale)
• ARCHI CLAUSOLA: Collega tutti i letterali nella stessa clausola
• ARCHI CONSISTENZA: Collega xᵢ con ¬xᵢ ovunque appaiono
• PARAMETRO: k = numero di clausole

CORRETTEZZA:
• ϕ soddisfacibile ⇒ scegli 1 letterale vero per clausola ⇒ insieme
indipendente di dimensione k
• Insieme indipendente di dim k ⇒ 1 vertice per clausola, consistenti ⇒
assegnamento soddisfacente
```

🔹 IndependentSet → VertexCover

```
COSTRUZIONE SEMPLICE:
F = "Su input ⟨G,k⟩:
   Restituisci ⟨G, |V|-k⟩"

CORRETTEZZA (FATTO MATEMATICO):
I è insieme indipendente ⟺ V\I è vertex cover
```

🔹 3SAT → 3-Coloring

```
COSTRUZIONE COMPLESSA:
• GADGET BASE: Triangolo con 3 colori base (TRUE, FALSE, BASE)
• GADGET VARIABILE: Forza xᵢ = TRUE o xᵢ = FALSE
• GADGET CLAUSOLA: Forza almeno un letterale TRUE
• CONNESSIONI: Propagano vincoli di consistenza
```

🎨 TECNICHE AVANZATE

🔹 Gadget Modulari

```
PRINCIPIO: Costruisci componenti riutilizzabili
APPLICAZIONI:
• Gadget per variabili: forzano scelte binarie
• Gadget per clausole: verificano soddisfacimento
• Gadget di connessione: propagano vincoli
```

🔹 Amplificazione di Parametri

```
TECNICA: Quando parametri non corrispondono naturalmente
ESEMPIO: Da problema con k a problema con f(k)
APPLICAZIONE: Duplicazione, moltiplicazione, offset
```

🔹 Riduzioni a Catena

```
STRATEGIA: A → B → C per semplificare costruzioni
ESEMPIO: 3SAT → IndSet → VertexCover → DominatingSet
VANTAGGIO: Riutilizza costruzioni esistenti
```

⚠️ ERRORI COMUNI E SOLUZIONI

🚫 ERRORI NELLE RIDUZIONI INDECIDIBILI

❌ Direzione Riduzione Sbagliata

```
ERRORE: Ridurre B ≤ₘ A invece di A ≤ₘ B
CORREZIONE: Per dimostrare B indecidibile, serve A ≤ₘ B con A indecidibile
```

❌ TM Ausiliaria Mal Costruita

```
ERRORE: M' non si comporta come previsto nei casi limite
CORREZIONE: Verificare sistematicamente ogni possibile input
```

❌ Correttezza Incompleta

```
ERRORE: Dimostrare solo (⇒) o solo (⇐)
CORREZIONE: SEMPRE entrambe le direzioni
```

🚫 ERRORI NELLE RIDUZIONI POLINOMIALI

❌ Riduzione Non Polinomiale

```
ERRORE: Costruzione richiede tempo esponenziale
CORREZIONE: Analizzare complessità di ogni passo
```

❌ Gadget Inconsistenti

```
ERRORE: Gadget non preservano soluzioni correttamente
CORREZIONE: Testare su esempi piccoli prima di generalizzare
```

❌ Parametri Mal Calcolati

```
ERRORE: k_target non corrisponde a k_source
CORREZIONE: Verificare mapping dimensioni soluzioni
```

📝 TEMPLATE MASTER

🔧 Template per Indecidibilità

```
PROBLEMA: Dimostrare che B è indecidibile

TEOREMA: B è indecidibile.

DIMOSTRAZIONE: Mostriamo ATM ≤ₘ B.

RIDUZIONE:
F = "Su input ⟨M,w⟩:
1. [COSTRUZIONE MACCHINA]
   Costruisci M':
   M' = "Su input x:
   [LOGICA SPECIFICA AL PROBLEMA]"
2. [OUTPUT]
   Restituisci [output appropriato]"

CORRETTEZZA:
• CALCOLABILITÀ: [argomento brevissimo]
• (⇒): ⟨M,w⟩ ∈ ATM ⟹ f(⟨M,w⟩) ∈ B
  [dimostrazione caso specifico]
• (⇐): f(⟨M,w⟩) ∈ B ⟹ ⟨M,w⟩ ∈ ATM
  [dimostrazione caso specifico]

CONCLUSIONE: ATM ≤ₘ B e ATM indecidibile, quindi B è indecidibile. □
```

🔧 Template per NP-Hard

```
PROBLEMA: Dimostrare che B è NP-hard

TEOREMA: B è NP-hard.

DIMOSTRAZIONE: Mostriamo 3SAT ≤ₚ B.

RIDUZIONE:
F = "Su input ϕ (formula 3CNF):
1. [GADGET CONSTRUCTION]
   [costruzione dettagliata]
2. [PARAMETER SETTING]
   [calcolo parametri]
3. [OUTPUT]
   Restituisci istanza di B"

CORRETTEZZA:
• POLINOMIALITÀ: [analisi complessità]
• (⇒): ϕ soddisfacibile ⟹ f(ϕ) ∈ B
  [costruzione esplicita soluzione]
• (⇐): f(ϕ) ∈ B ⟹ ϕ soddisfacibile
  [estrazione esplicita assegnamento]

CONCLUSIONE: 3SAT ≤ₚ B e 3SAT è NP-hard, quindi B è NP-hard. □
```

🎯 STRATEGIA DI SCELTA

📊 Decision Tree per Problema Sorgente

Per Indecidibilità:

```
ATM se: problema generale sui linguaggi TM
HALTTM se: problema specifico su terminazione
ETM se: problema su linguaggi vuoti
REGULARTM se: problema su proprietà strutturali
```

Per NP-Hard:

```
3SAT se: problema logico/combinatorio generale
IndependentSet se: problema su grafi (selezione)
```

```
VertexCover se: problema su grafi (copertura)
HamiltonianCycle se: problema su ordinamenti/percorsi
```

🧭 Heuristic per Costruzioni

1. **Inizia semplice**: prova prima costruzioni dirette
2. **Usa gadget**: per problemi complessi, costruisci modulari
3. **Testa piccolo**: verifica correttezza su esempi minimali
4. **Scala gradualmente**: espandi a caso generale
5. **Verifica entrambe le direzioni**: non dimenticare (⇐)

✅ CHECKLIST FINALE

Per ogni riduzione:

- [ ] Problema sorgente scelto appropriatamente
- [ ] Funzione di riduzione definita costruttivamente
- [ ] Calcolabilità/polinomialità dimostrata
- [ ] Entrambe le direzioni della correttezza provate
- [ ] Costruzioni esplicite per mappare soluzioni
- [ ] Casi limite considerati
- [ ] Conclusione di difficoltà derivata correttamente