Lezione di Informatica Teorica: Riduzioni e Indecidibilità del PCP

Appunti da Trascrizione Automatica

30 giugno 2025

Indice

1 Introduzione alle Riduzioni

Una *riduzione* è uno strumento fondamentale in Teoria della Computazione per dimostrare la difficoltà computazionale di problemi.

Definizione 1 (Riduzione Formale)**.** *Siano A e B due linguaggi (problemi di decisione). Una funzione $f : \Sigma^* \to \Sigma^*$ è una **riduzione calcolabile** di A a B (indicata $A \leq_m B$) se:*

1. *f è calcolabile (esiste una Macchina di Turing che calcola f in tempo finito per ogni input).*
2. *Per ogni stringa $w \in \Sigma^*$, si ha che $w \in A \iff f(w) \in B$.*

Il significato di $w \in A \iff f(w) \in B$ è duplice:

- *Se $w \in A$, allora $f(w) \in B$.*
- *Se $w \notin A$, allora $f(w) \notin B$.*

1.1 Scopo delle Riduzioni

Le riduzioni sono utilizzate per dimostrare l'indecidibilità di problemi.

Teorema 1. *Se $A \leq_m B$ e A è un problema indecidibile, allora anche B è un problema indecidibile.*

Dimostrazione. Supponiamo per contraddizione che B sia decidibile. Allora esisterebbe una Macchina di Turing M_B che decide B. Poiché f è calcolabile, possiamo costruire una Macchina di Turing M_A che decide A come segue:

1. Input w.
2. Calcola $f(w)$.
3. Simula M_B su $f(w)$.
4. Accetta se M_B accetta, rifiuta se M_B rifiuta.

Per la definizione di riduzione, $w \in A \iff f(w) \in B$. Quindi M_A decide A. Ma questo contraddice l'ipotesi che A sia indecidibile. Dunque, B deve essere indecidibile. □

Teorema 2 (Transitività delle Riduzioni)**.** *Se $A \leq_m B$ e $B \leq_m C$, allora $A \leq_m C$.*

Dimostrazione. Siano f la riduzione da A a B e g la riduzione da B a C. Entrambe f e g sono calcolabili. Allora la composizione $g \circ f$ è anch'essa calcolabile. Inoltre, $w \in A \iff f(w) \in B \iff g(f(w)) \in C$. Quindi $g \circ f$ è una riduzione calcolabile da A a C. □

2 Problema di Corrispondenza di Post (PCP)

Il Problema di Corrispondenza di Post (PCP) è un problema di decisione sulle stringhe.

Definizione 2 (PCP)**.** *Un'istanza del PCP è data da due liste non-vuote di stringhe, $A = (A_1, A_2, \ldots, A_k)$ e $B = (B_1, B_2, \ldots, B_k)$, definite su un alfabeto Σ. Il problema chiede se esista una sequenza di indici non-vuota $I = (i_1, i_2, \ldots, i_m)$ con $m \geq 1$ e $1 \leq i_j \leq k$ per ogni j, tale che:*

$$A_{i_1} A_{i_2} \ldots A_{i_m} = B_{i_1} B_{i_2} \ldots B_{i_m}$$

La risposta è "sì" se una tale sequenza esiste, "no" altrimenti.

Esempio 1. *Consideriamo le liste:* $A = (A_1 = 1, A_2 = 10111, A_3 = 10)$ $B = (B_1 = 111, B_2 = 10, B_3 = 0)$

Una possibile sequenza di indici è $I = (2, 1, 1, 3)$*:* $A_2A_1A_1A_3 = (10111)(1)(1)(10) = 101111110$ $B_2B_1B_1B_3 = (10)(111)(111)(0) = 101111110$ *Poiché le stringhe risultanti sono uguali, questa istanza ha una soluzione ("sì").*

2.1 Proprietà del PCP

- Il PCP è un problema **ricorsivamente enumerabile** (in R). Possiamo ideare un algoritmo che, data un'istanza, cerca tutte le possibili sequenze di indici di lunghezza crescente. Se trova una soluzione, accetta. Se non trova una soluzione in un tempo finito, potrebbe continuare a cercare indefinitamente.
- Il PCP è **indecidibile** (non in R). Non esiste un algoritmo che, per ogni istanza del PCP, termina e fornisce la risposta corretta ("sì" o "no"). Dimostreremo questo usando le riduzioni.

3 Problema di Corrispondenza di Post Modificato (MPCP)

Per dimostrare l'indecidibilità del PCP, useremo un problema intermedio, il MPCP.

Definizione 3 (MPCP)**.** *Un'istanza del MPCP è data da due liste non-vuote di stringhe,* $A = (A_1, A_2, \ldots, A_k)$ *e* $B = (B_1, B_2, \ldots, B_k)$*, definite su un alfabeto* Σ*. Il problema chiede se esista una sequenza di indici* $I = (i_1, i_2, \ldots, i_m)$ *con* $m \geq 1$ *e* $1 \leq i_j \leq k$ *per ogni* j*, tale che:*

1. *La sequenza deve iniziare con l'indice* 1*:* $i_1 = 1$*.*
2. $A_{i_1}A_{i_2}\ldots A_{i_m} = B_{i_1}B_{i_2}\ldots B_{i_m}$

La risposta è "sì" se una tale sequenza esiste, "no" altrimenti.

L'unica differenza tra PCP e MPCP è la condizione che la sequenza di indici debba iniziare per 1.

4 Riduzione 1: $LU \leq_m MPCP$

Dimostriamo che il Linguaggio Universale (LU) si riduce al MPCP. Dato che LU è indecidibile, questo implicherà che MPCP è indecidibile.

4.1 Idea della Riduzione

Un'istanza di LU è una coppia (M, w), dove M è una Macchina di Turing e w è una stringa. Un'istanza di MPCP è una coppia di liste di stringhe (A, B). La funzione di riduzione f deve trasformare (M, w) in (A, B). L'idea centrale è simulare la computazione di M su w usando le stringhe delle liste A e B. La computazione di una MT può essere rappresentata come una sequenza di *descrizioni istantanee* (ID) separate da un simbolo speciale (e.g., #). $ID_1\#ID_2\#ID_3\#\ldots\#ID_k$ Vogliamo costruire le liste A e B in modo tale che, se M accetta w, allora esista una sequenza di indici per MPCP che costruisce la stessa stringa. Le stringhe di B saranno sempre "un passo avanti" rispetto alle stringhe di A, simulando la prossima ID. Quando M accetta, le stringhe di A avranno la possibilità di "recuperare" e allinearsi con quelle di B.

4.2 Costruzione delle Liste A e B

L'alfabeto del MPCP sarà $\Gamma \cup Q \cup \{\#, \$, *\}$, dove Γ è l'alfabeto del nastro di M, Q è l'insieme degli stati di M, e $\#, \$, *$ sono nuovi simboli. Assumiamo che M non scriva mai il simbolo blank e che il suo nastro sia semi-infinito a destra (non si muova mai a sinistra della posizione iniziale). Queste sono assunzioni standard che non limitano la generalità delle MT.

Le liste A e B sono costruite con le seguenti classi di coppie di stringhe (A_i, B_i):

1. **Coppia Iniziale (obbligatoria per MPCP):** Questa coppia inizia la simulazione e garantisce che B sia un passo avanti.
   - $A_1 = \#$
   - $B_1 = \#q_0w\#$ (dove q_0 è lo stato iniziale di M, w è la stringa d'input, e # è un simbolo di confine).
2. **Coppie di Copia:** Permettono di copiare simboli di configurazione che non sono sotto la testina. Per ogni simbolo $x \in \Gamma \cup \{\#\}$:
   - $A_i = x$
   - $B_i = x$
3. **Coppie di Transizione:** Simulano il movimento della testina e la modifica del nastro secondo le regole di transizione di M. Queste regole sono generate per ogni $q \in Q \setminus F$ (stato non finale) e ogni $X \in \Gamma \cup \{\#\}$.
   - **Spostamento a destra (R):** Se $\delta(q, X) = (p, Y, R)$:
     - $A_i = qX$
     - $B_i = Yp$

     (Esempio: se M legge X nello stato q, scrive Y e va nello stato p muovendosi a destra, allora qX in A corrisponde a Yp in B). *Caso speciale: X è un blank (simbolo di bordo #).* Se $\delta(q, \#) = (p, Y, R)$:
     - $A_i = q\#$
     - $B_i = Yp\#$
   - **Spostamento a sinistra (L):** Se $\delta(q, X) = (p, Y, L)$:
     - $A_i = ZqX$ (per ogni $Z \in \Gamma \cup \{\#\}$)
     - $B_i = pZY$

     (Esempio: se M legge X nello stato q, scrive Y e va nello stato p muovendosi a sinistra, allora la sequenza ZqX in A (dove Z è il simbolo a sinistra di q) corrisponde a pZY in B).
4. **Coppie di Accettazione/Recupero:** Queste coppie sono utilizzate solo quando M entra in uno stato accettante $q_f \in F$. Permettono alla stringa concatenata di A di "recuperare" la lunghezza della stringa concatenata di B. Per ogni $q_f \in F$ e ogni $X, Y \in \Gamma$:
   - $A_i = Xq_fY$
   - $B_i = q_fY$

   Questo riduce la differenza di lunghezza tra A_i e B_i per q_f e Y.
   - $A_i = Xq_f\#$

- $B_i = q_f\#$

- $A_i = q_f Y\#$
- $B_i = q_f\#$

- $A_i = q_f\#\#$
- $B_i = q_f\#$

(Il professore ha menzionato $XQY \to QY$ come regola generale per $Q \in F$. Il principio è che la stringa di A diventa più lunga o uguale, permettendo di chiudere la partita.)

5. **Coppia Finale:** Permette di completare il match una volta che la MT è in uno stato accettante e la differenza di lunghezza è stata recuperata. Per ogni $q_f \in F$:

   - $A_i = q_f\#\#$ (due simboli # per A)
   - $B_i = \#\#$ (due simboli # per B)

4.3 Esempio di Simulazione

Sia M la Macchina di Turing definita come segue (dal diagramma della lezione): Stati: $Q = \{q_1, q_2, q_3\}$ (dove q_1 è iniziale, q_3 è finale). Alfabeto del nastro: $\Gamma = \{0, 1, \beta\}$ (β è il simbolo blank). Funzione di transizione (interpretata dal diagramma): $\delta(q_1, 0) = (q_2, 1, R)$ $\delta(q_1, 1) = (q_1, 0, L)$ $\delta(q_2, 1) = (q_1, 0, R)$ $\delta(q_1, \beta) = (q_2, 1, L)$ (loop dal blank, torna indietro e cambia stato) $\delta(q_2, \beta) = (q_3, 0, R)$ (accetta e si sposta a destra)

Sia la stringa d'input $w = 01$. La computazione di M su $w = 01$ è: 1. $\#q_1 01\#$ (Configurazione iniziale) 2. $\#1q_2 1\#$ (Da q_1 legge 0, scrive 1, va in q_2, si sposta a R) 3. $\#10q_1\#$ (Da q_2 legge 1, scrive 0, va in q_1, si sposta a R) 4. $\#1q_2 1\#$ (Da q_1 legge β, scrive 1, va in q_2, si sposta a L) 5. $\#10q_3\#$ (Da q_2 legge β, scrive 0, va in q_3, si sposta a R) - Stato accettante q_3.

Come la riduzione costruisce la soluzione MPCP:

- Si inizia con la coppia iniziale: $A_1 = \#$, $B_1 = \#q_1 01\#$. La stringa concatenata di A (finora #) è "dietro" quella di B (finora $\#q_1 01\#$).

- Per simulare $\#q_1 01\# \to \#1q_2 1\#$:

  - Usiamo coppie di copia per # iniziale: $(A_i = \#, B_i = \#)$.
  - Per la transizione $q_1 0 \to 1q_2$: usiamo la coppia $(A_i = q_1 0, B_i = 1q_2)$.
  - Usiamo coppie di copia per 1#: $(A_i = 1, B_i = 1)$ e $(A_i = \#, B_i = \#)$.

  A questo punto, la stringa concatenata di A sarà $\#q_1 01\#$, mentre quella di B sarà $\#q_1 01\#\#1q_2 1\#$. B è ancora un passo avanti.

- Questo processo continua. Le regole di copia e transizione assicurano che la stringa costruita da B contenga la sequenza di ID, con la stringa costruita da A che la segue con un ID di ritardo.

- Quando la computazione raggiunge uno stato finale (es. q_3 nell'esempio, $ID_5 = \#10q_3\#$), le coppie di accettazione (Classe 4) permettono alla stringa di A di "catturare" la stringa di B. Ad esempio, per $Xq_f Y \to q_f Y$, A contribuisce con 3 simboli e B con 2, riducendo il gap.

- Infine, la coppia finale (Classe 5) permette di pareggiare completamente le stringhe. Se A arriva a $\#ID_1\#ID_2\#\ldots\#ID_k\#q_f\#\#$ e B arriva a $\#ID_1\#ID_2\#\ldots\#ID_k\#\#\#$, allora la coppia finale farà sì che A e B diventino uguali.

4.4 Dimostrazione

Sketch di Dimostrazione 1. *Si deve dimostrare che (M, w) è un'istanza "sì" di LU se e solo se l'istanza (A, B) generata dalla riduzione è un'istanza "sì" di MPCP.*

Parte 1: Se M accetta w, allora l'istanza MPCP ha soluzione. *Se M accetta w, significa che esiste una sequenza di configurazioni $ID_1, ID_2, \ldots, ID_k$ dove $ID_1 = q_0w$, e ID_k è una configurazione accettante. Costruiamo la sequenza di indici per MPCP come segue: Iniziamo sempre con l'indice 1 (la coppia iniziale). Questo produce $A_{concat} = \#$ e $B_{concat} = \#ID_1\#$. Poi, si scelgono le coppie di stringhe (A_i, B_i) dalle classi 2 e 3 che simulano la transizione da ID_j a ID_{j+1}. Ogni volta che si concatena una coppia di transizione, la stringa di A "completa" la ID_j e la stringa di B "inizia" la ID_{j+1}. Questo fa sì che A_{concat} diventi $\#ID_1\#ID_2\#\ldots\#ID_{k-1}\#$ e B_{concat} diventi $\#ID_1\#ID_2\#\ldots\#ID_k\#$. Poiché ID_k è una configurazione accettante, possiamo usare le coppie di Classe 4 per "rimuovere" i simboli vicino allo stato finale in A e B in modo disuguale (o meglio, A apporta più simboli di B per la stessa parte di configurazione), riducendo il divario di lunghezza. Infine, la coppia di Classe 5 permette di far terminare le stringhe con esattamente gli stessi simboli, facendo sì che $A_{concat} = B_{concat}$. Dunque, esiste una soluzione per l'istanza MPCP.*

Parte 2: Se l'istanza MPCP ha soluzione, allora M accetta w. *Supponiamo che l'istanza MPCP (A, B) generata dalla riduzione abbia una soluzione. Per la definizione di MPCP, questa soluzione deve iniziare con la coppia (A_1, B_1), che è $(\#, \#q_0w\#)$. L'unico modo per le stringhe concatenate di A e B di rimanere allineate e infine eguagliarsi è che le coppie usate simulino correttamente le transizioni di M. Qualsiasi sequenza di indici che non rispetti la simulazione delle transizioni (ad esempio, usando una coppia (X, Y) senza che Y sia il risultato della transizione di X) non permetterebbe mai alle stringhe di allinearsi correttamente a causa dell'alternanza di simboli di nastro e stato. Poiché le stringhe di B sono sempre un passo avanti (contengono $\#ID_j\#ID_{j+1}\#$ mentre A ha solo $\#ID_j\#$), l'unico modo per A di "recuperare" e far sì che le stringhe concatenate di A e B diventino uguali è tramite l'uso delle coppie di Classe 4 (Accettazione/Recupero) e Classe 5 (Finale). Le coppie di Classe 4 e 5 possono essere utilizzate solo se la configurazione attuale di M (simulata) contiene uno stato accettante. Pertanto, se esiste una soluzione MPCP, la simulazione deve aver raggiunto una configurazione accettante, il che significa che M accetta w.*

Poiché *LU* è indecidibile e $LU \leq_m MPCP$, concludiamo che *MPCP* è **indecidibile**.

5 Riduzione 2: $MPCP \leq_m PCP$

Ora dimostriamo che *MPCP* si riduce al *PCP*. Dato che *MPCP* è indecidibile, questo implicherà che *PCP* è indecidibile.

5.1 Idea della Riduzione

Un'istanza di MPCP è una coppia di liste (A, B). Un'istanza di PCP è una coppia di liste (C, D). La funzione di riduzione f deve trasformare (A, B) in (C, D). La sfida è che PCP non richiede che la soluzione inizi con un indice specifico, mentre MPCP sì (indice 1). Dobbiamo modificare le

liste (A, B) in (C, D) in modo che qualsiasi soluzione PCP debba iniziare con la coppia modificata dall'indice 1 dell'MPCP.

5.2 Costruzione delle Liste C e D

Introduciamo due nuovi simboli non presenti nell'alfabeto originale: $*$ (asterisco) e \$ (dollaro).

Le liste C e D sono costruite come segue:

1. **Trasformazione delle Coppie Originali:** Per ogni coppia (A_i, B_i) con $i \in \{1, \ldots, k\}$:
   - C_i: Ogni simbolo di A_i è seguito da un asterisco. Esempio: se $A_i = s_1 s_2 \ldots s_m$, allora $C_i = s_1 * s_2 * \ldots s_m *$.
   - D_i: Ogni simbolo di B_i è preceduto da un asterisco. Esempio: se $B_i = t_1 t_2 \ldots t_n$, allora $D_i = *t_1 * t_2 \cdots * t_n$.
2. **Coppia Iniziale Forzata:** Per forzare la sequenza a iniziare con l'equivalente dell'indice 1 di MPCP:
   - $C_0 = *C_1$ (la stringa C_1 (ottenuta dal A_1 originale) con un asterisco aggiunto all'inizio). Esempio: se $A_1 = 1$, $C_1 = 1*$. Allora $C_0 = *1*$.
   - $D_0 = D_1$ (la stringa D_1 (ottenuta dal B_1 originale)). Esempio: se $B_1 = 111$, $D_1 = *1 * 1 * 1$. Allora $D_0 = *1 * 1 * 1$.

   Nota: Tutte le stringhe in C_i (per $i > 0$) iniziano con un simbolo dell'alfabeto originale seguito da *, mentre tutte le stringhe in D_i iniziano con *. L'unica eccezione è C_0 che inizia con *. Questo forza la scelta di C_0 e D_0 come prima coppia della soluzione PCP.
3. **Coppia Finale Forzata:** Per garantire che le stringhe concatenate possano eguagliarsi alla fine:
   - $C_{k+1} =$ \$ (un singolo simbolo dollaro)
   - $D_{k+1} = *$\$ (asterisco seguito da dollaro)

   Questo assicura che se le stringhe si allineano in modo corretto con gli asterischi e poi matchano con il dollaro, allora la soluzione è stata trovata.

5.3 Esempio di Costruzione

Partiamo dall'istanza PCP dell'Esempio iniziale, ma trattandola come un'istanza MPCP (dove l'indice 1 deve essere il primo): $A = (A_1 = 1, A_2 = 10111, A_3 = 10)$ $B = (B_1 = 111, B_2 = 10, B_3 = 0)$

Applichiamo la riduzione per ottenere (C, D):

- **Coppie trasformate (da indice 1 a 3):** $C_1 = 1*$ $D_1 = *1 * 1 * 1$

  $C_2 = 1 * 0 * 1 * 1 * 1*$ $D_2 = *1 * 0$

  $C_3 = 1 * 0*$ $D_3 = *0$
- **Coppia Iniziale Forzata (indice 0):** $C_0 = *1*$ $D_0 = *1 * 1 * 1$
- **Coppia Finale Forzata (indice 4):** $C_4 =$ \$ $D_4 = *$\$

L'istanza PCP risultante è $C = (C_0, C_1, C_2, C_3, C_4)$ e $D = (D_0, D_1, D_2, D_3, D_4)$.

5.4 Dimostrazione

Sketch di Dimostrazione 2. *Si deve dimostrare che l'istanza (A, B) è "sì" per MPCP se e solo se l'istanza (C, D) generata dalla riduzione è "sì" per PCP.*

Parte 1: Se (A, B) ha soluzione MPCP, allora (C, D) ha soluzione PCP. *Se (A, B) ha una soluzione MPCP, esiste una sequenza di indici $I = (1, i_2, \ldots, i_m)$ tale che $A_1 A_{i_2} \ldots A_{i_m} = B_1 B_{i_2} \ldots B_{i_m}$. Costruiamo la soluzione PCP come segue: $I' = (0, 1, i_2, \ldots, i_m, k+1)$. La stringa C_{concat} sarà $C_0 C_1 C_{i_2} \ldots C_{i_m} C_{k+1}$. La stringa D_{concat} sarà $D_0 D_1 D_{i_2} \ldots D_{i_m} D_{k+1}$.*

*Sostituendo le definizioni: $C_{concat} = (*A'_1)(A'_1 \text{ con } * \text{dopo})(A'_{i_2} \text{ con } * \text{dopo}) \ldots (A'_{i_m} \text{ con } * \text{dopo})(\$)$ $D_{concat} = (B'_1 \text{ con } * \text{prima})(B'_1 \text{ con } * \text{prima})(B'_{i_2} \text{ con } * \text{prima}) \ldots (B'_{i_m} \text{ con } * \text{prima})(*\$)$*

Se $A_1 A_{i_2} \ldots A_{i_m} = B_1 B_{i_2} \ldots B_{i_m}$, allora è facile vedere che: $A'_1 A'_{i_2} \ldots A'_{i_m}$ con asterischi finali sarà uguale a $B'_1 B'_{i_2} \ldots B'_{i_m}$ con asterischi iniziali. L'aggiunta iniziale di $C_0 = *A'_1$ e $D_0 = B'_1$ (dove B'_1 ha l'asterisco iniziale) fa sì che il primo asterisco di C_0 si allinei con il primo asterisco di D_0. Poi, la corrispondenza simbolo-asterisco e asterisco-simbolo si mantiene. L'aggiunta finale di \$ e $\$$ permette di concludere il match.*

Parte 2: Se (C, D) ha soluzione PCP, allora (A, B) ha soluzione MPCP. *Supponiamo che (C, D) abbia una soluzione PCP, ovvero una sequenza di indici $I' = (j_1, j_2, \ldots, j_p)$ tale che $C_{j_1} C_{j_2} \ldots C_{j_p} = D_{j_1} D_{j_2} \ldots D_{j_p}$.*

- ***Forzatura dell'inizio:*** *L'unica stringa in C che inizia con un asterisco è C_0. Tutte le stringhe in D (tranne $D_{k+1} = *\$$) iniziano con un asterisco. Pertanto, per un match, la prima coppia usata deve essere (C_0, D_0). Questo garantisce che $j_1 = 0$.*
- ***Allineamento degli asterischi:*** *Dopo l'inizio con (C_0, D_0), tutte le stringhe di C_i (per $i > 0$) terminano con un asterisco, e tutte le stringhe di D_i iniziano con un asterisco. Questo significa che un C_i e un D_j possono iniziare a corrispondere solo se l'ultimo carattere del precedente segmento di C è un simbolo non-asterisco, e il primo carattere del successivo segmento di D è un asterisco. Questo meccanismo di "passo-passo" impone che la sequenza di indici debba essere tale da mantenere una perfetta alternanza di simboli normali e asterischi per tutta la stringa.*
- ***Forzatura della fine:*** *Per terminare la stringa con un match, deve essere usata la coppia $(C_{k+1}, D_{k+1}) = (\$, *\$)$. Questo assicura che il match si concluda con un dollaro.*

Data la struttura della riduzione, se le stringhe concatenate di C e D sono uguali, significa che la sequenza originale di A (senza asterischi e dollari) deve essere uguale alla sequenza originale di B (senza asterischi e dollari). Poiché la soluzione PCP deve iniziare con (C_0, D_0), questo implica che la soluzione MPCP inizia con la coppia (A_1, B_1), soddisfacendo il requisito di MPCP. Dunque, esiste una soluzione per l'istanza MPCP.

Poiché $MPCP$ è indecidibile e $MPCP \leq_m PCP$, concludiamo che PCP è **indecidibile**.