Riassunto delle cose utili - Secondo parziale

Gabriel Rovesti

Indice

1 Introduzione

Questo documento presenta una raccolta sistematizzata delle principali tecniche utilizzate nella seconda parte del corso sui linguaggi formali, con particolare attenzione a:

- Dimostrare se un linguaggio è decidibile
- Dimostrare se un linguaggio è indecidibile
- Dimostrare se un linguaggio è NP-hard/NP-completo
- Varianti delle macchine di Turing e loro equivalenza

Per ogni categoria, verranno presentati i principi teorici, le tecniche specifiche con esempi di applicazione, e schemi risolutivi riutilizzabili.

2 Dimostrare se un linguaggio è decidibile

2.1 Costruzione diretta di un decisore

La tecnica più diretta consiste nel costruire esplicitamente una macchina di Turing che decide il linguaggio.

Schema generale:

1. Definire l'algoritmo di decisione
2. Dimostrare che l'algoritmo termina sempre
3. Dimostrare che l'algoritmo accetta esattamente le stringhe del linguaggio

Esempio: $COMPLEMENT_{DFA} = \{(A, B) \mid A$ e B sono DFA e $L(A) = \overline{L(B)}\}$

Soluzione: Costruiamo un decisore N che:

1. Costruisce il DFA C che accetta l'intersezione dei linguaggi di A e B
2. Esegue M su input (C). Se M accetta, rifiuta; se M rifiuta, accetta

Poiché l'intersezione di due DFA è costruibile in tempo finito e E_{DFA} è decidibile, N termina sempre e decide correttamente $COMPLEMENT_{DFA}$.

2.2 Riduzione a problemi decidibili noti

Se possiamo ridurre il nostro problema a un problema noto decidibile, allora anche il nostro problema è decidibile.

Esempio: $EQ_{DFA,REX} = \{(D, R) \mid D$ è un DFA, R è un'espressione regolare e $L(D) = L(R)\}$

Soluzione: Costruiamo un decisore N che:

1. Converte R in un DFA equivalente D_R
2. Esegue M su input (D, D_R) e ritorna lo stesso risultato di M

Poiché ogni espressione regolare può essere convertita in un DFA equivalente e EQ_{DFA} è decidibile, anche $EQ_{DFA,REX}$ è decidibile.

2.3 Enumeratori e ordinamento standard

Un linguaggio è decidibile se e solo se esiste un enumeratore che lo enumera seguendo l'ordinamento standard delle stringhe.

Teorema: Un linguaggio L è decidibile se e solo se L è Turing-riconoscibile ed esiste un enumeratore E che enumera L in ordine lessicografico.

Dimostrazione:

- ($\Rightarrow$) Se L è decidibile, esiste una TM M che lo decide. Possiamo costruire un enumeratore E che genera tutte le stringhe in ordine lessicografico, le testa con M e produce quelle accettate.
- ($\Leftarrow$) Se esiste un enumeratore E che enumera L in ordine standard, possiamo costruire un decisore D per L: Su input w, D simula E. Se E produce w, D accetta. Se E produce una stringa lessicograficamente maggiore di w, D rifiuta. D termina sempre perché E enumera le stringhe in ordine.

3 Dimostrare se un linguaggio è indecidibile

3.1 Riduzione dal problema dell'arresto

Il metodo principale per dimostrare che un linguaggio è indecidibile è la riduzione da $A_{TM} = \{(M, w) \mid M \text{ accetta } w\}$.

Schema generale per la riduzione:

1. Assumere per assurdo che il linguaggio L sia decidibile
2. Costruire un decisore S per A_{TM} usando il decisore ipotetico R per L
3. Poiché A_{TM} è indecidibile, concludere che L non può essere decidibile

Esempio: $A_{1010} = \{M \mid M \text{ è una TM tale che } 1010 \in L(M)\}$

Dimostrazione: Supponiamo per assurdo che A_{1010} sia decidibile con decisore R. Costruiamo un decisore S per A_{TM}:

$S =$ "Su input (M, w):

1. Costruisci la seguente macchina M_w:
   - $M_w =$ "Su input x:
     (a) Se $x \neq 1010$, rifiuta
     (b) Se $x = 1010$, esegui M su input w
     (c) Se M accetta, accetta
     (d) Se M rifiuta, rifiuta"
2. Restituisci (M_w)"

S calcola una riduzione da A_{TM} ad A_{1010}: $(M, w) \in A_{TM}$ se e solo se $M_w \in A_{1010}$.

3.2 Varianti del problema dell'arresto

Esempio: EVEN-HALTS $= \{M \mid \text{per ogni numero naturale } n \text{ pari, } M \text{ termina la computazione su } n\}$

Dimostrazione: Riduciamo A_{TM} a EVEN-HALTS. Costruiamo una funzione f che mappa (M, w) a M', dove M' è definita come:

$M' =$ "Su input n:

1. Se n è dispari, entra in un loop infinito
2. Se n è pari, simula M su input w
3. Se M si ferma su w, M' si ferma
4. Se M non si ferma su w, M' entra in un loop infinito"

$(M, w) \in A_{TM}$ se e solo se $M' \in$ EVEN-HALTS.

3.3 Tecniche specifiche di riduzione

Esempio: MAGIC$_TM$ Consideriamo una TM M che scrive "xyzzy" su cinque celle adiacenti del nastro, assumendo che l'alfabeto di input non contenga i simbololi x, y, z.

Dimostrazione: Riduciamo A_{TM} a $MAGIC_TM$ costruendo M' che:

1. Verifica che i simbololi x, y, z non compaiano in w, né nell'alfabeto di input o nell'alfabeto del nastro di M
2. Costruisce una macchina M' che simula l'esecuzione di M su input w senza usare i simboli x, y, z
3. Se M accetta, scrive xyzzy sul nastro, altrimenti rifiuta senza modificare il nastro

4 Dimostrare se un linguaggio è NP-hard/NP-completo

4.1 Struttura delle dimostrazioni NP-complete

Per dimostrare che un problema è NP-completo, dobbiamo dimostrare due cose:

1. Il problema è in NP (esiste un verificatore polinomiale)
2. Il problema è NP-hard (esiste una riduzione polinomiale da un problema NP-hard noto)

4.2 Verificatori polinomiali

Schema generale: Un verificatore polinomiale V per un linguaggio L prende in input una stringa w e un certificato c, e:

1. Verifica in tempo polinomiale che c ha le proprietà richieste
2. Accetta se e solo se $w \in L$ e c è un certificato valido per w

Esempio: HAM375 $= \{G \mid G$ è un grafo con n vertici che ha un ciclo che attraversa esattamente $n - 375$ vertici$\}$

Verificatore: Input: (G, C), dove G è un grafo e C è un ciclo proposto

1. Verifica che C contenga esattamente $n - 375$ vertici di G
2. Verifica che ogni vertice in C appaia esattamente una volta
3. Verifica che per ogni coppia di vertici consecutivi in C, esista un arco in G che li collega
4. Se tutte le condizioni sono soddisfatte, accetta. Altrimenti, rifiuta

4.3 Riduzioni polinomiali

Schema generale: Per dimostrare che $A \leq_p B$, costruiamo una funzione f calcolabile in tempo polinomiale tale che: $w \in A$ se e solo se $f(w) \in B$

Esempio: 3-COLOR $\leq_p$ LIST-COLORING Data un'istanza $G = (V, E)$ di 3-COLOR, costruiamo un'istanza (G', L) di LIST-COLORING:

- $G' = G$
- Per ogni vertice $v \in V$, $L(v) = \{1, 2, 3\}$

Chiaramente, G ha una 3-colorazione se e solo se (G', L) ha una colorazione valida per LIST-COLORING.

4.4 Riduzioni da problemi classici

Da HAMILTON a PebbleDestruction: Data un'istanza $G = (V, E)$ del Circuito Hamiltoniano, costruiamo un'istanza di PebbleDestruction:

- Il grafo è $G' = G$
- Per ogni vertice $v \in V$, $p(v) = 2$

Se G ha un circuito Hamiltoniano, allora esiste una sequenza di mosse in G' che rimuove tutti i ciottoli tranne uno seguendo il circuito. Viceversa, se esiste una soluzione per PebbleDestruction in G', la sequenza di mosse corrisponde a un cammino che visita ogni vertice esattamente una volta.

Da VERTEX-COVER a SUPPLY: Sia $f((G, k)) = (S_1, \ldots, S_n, k)$ dove:

- n è il numero di vertici di G
- $S_i[j] = 1$ se (i, j) è un arco di G, $S_i[j] = 0$ altrimenti

Se $(G, k) \in$ VERTEX-COVER, esiste un vertex cover C di dimensione k, e prendendo la fornitura $T = C$ abbiamo che $\sum_{j \in T} S_i[j] = 1$ per ogni vertice i.

5 Varianti delle macchine di Turing

5.1 Equivalenza tra varianti

Tutte le seguenti varianti di macchine di Turing riconoscono esattamente la classe dei linguaggi Turing-riconoscibili.

5.1.1 Macchine di Turing con reset a sinistra

Una macchina di Turing con reset a sinistra ha la funzione di transizione:

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R, RESET\}$$

Il simbolo $RESET$ riporta la testina alla prima cella del nastro.

Equivalenza: Una TM standard può simulare una TM con reset mantenendo traccia della posizione corrente e implementando il reset come una sequenza di mosse a sinistra.

5.1.2 Tag-Turing machine

Una tag-Turing machine ha una testina di lettura che può solo spostarsi a destra, e una testina di scrittura che può spostarsi sia a sinistra che a destra.

Equivalenza: Una TM standard può simulare una tag-Turing machine usando due nastri: uno per simulare il nastro di lettura e uno per il nastro di scrittura.

5.1.3 Macchine di Turing bidimensionali

Una TM bidimensionale utilizza una griglia bidimensionale infinita di celle come nastro, con transizioni della forma:

$$\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{\hat{}, \_, \text{¸}, \text{ß}\}$$

Equivalenza: Una TM standard può simulare una TM bidimensionale usando una codifica della griglia su un nastro unidimensionale, memorizzando le coordinate correnti e navigando la rappresentazione codificata.

5.1.4 Macchine di Turing a testine multiple

Una TM a k testine ha la funzione di transizione:

$$\delta : Q \times \Gamma^k \mapsto Q \times \Gamma^k \times \{L, R\}^k$$

Equivalenza: Una TM standard può simulare una TM a testine multiple usando $k + 1$ tracce sul nastro: una per il contenuto originale e k per marcare le posizioni delle testine.

5.2 Varianti con funzionalità aggiuntive

5.2.1 Macchine di Turing con inserimento

Hanno la funzione di transizione $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R, I\}$ dove I indica l'inserimento di una nuova cella.

Equivalenza: Possono essere simulate da TM standard usando un simbolo speciale per marcare le celle inserite e riorganizzando il contenuto quando necessario.

5.2.2 Macchine di Turing con stack di nastri

Possiedono due azioni aggiuntive: Push (salvare l'intero nastro inserendolo in uno stack) e Pop (ripristinare l'ultimo nastro salvato dallo stack).

Equivalenza: Una TM multinastro può simulare questa variante usando nastri aggiuntivi per implementare lo stack di configurazioni.

5.2.3 Macchine di Turing ad albero binario

Usano un albero binario infinito come nastro, con transizioni $\delta : Q \times \Gamma \mapsto Q \times \Gamma \times \{P, L, R\}$ dove $P =$ padre, $L =$ figlio sinistro, $R =$ figlio destro.

Equivalenza: Una TM standard può simulare questa variante codificando l'albero usando sequenze di L e R per rappresentare i percorsi dalla radice.

5.3 Varianti specializzate

5.3.1 Macchine di Turing ecologiche (ETM)

Possono spostarsi a sinistra (L), a destra (R) o passare all'altro lato del nastro (F).

Equivalenza: Una TM standard a due nastri può simulare una ETM usando un nastro per il lato "fronte" e uno per il lato "retro".

5.3.2 R2-L3 Turing Machine

Può effettuare solo due mosse consecutive a destra (R2) oppure tre mosse a sinistra (L3).

Equivalenza: Una TM standard può simulare questa variante tenendo traccia delle mosse effettuate e simulando le sequenze R2 o L3 come operazioni atomiche.

5.3.3 Macchine con Copy-Paste (CPTM)

Hanno operazioni aggiuntive per selezionare e copiare porzioni di nastro: C (inizio selezione), V (fine selezione), P (incolla).

Equivalenza: Una TM standard può simulare le CPTM usando nastri aggiuntivi per memorizzare le selezioni e il contenuto copiato.

5.3.4 Macchine Stateless (JTM)

Hanno un solo stato e la funzione di transizione $\delta : \Gamma \mapsto \Gamma \times \{L, R\}$.

Equivalenza: Le JTM possono simulare TM standard codificando gli stati sul nastro e usando una codifica appropriata delle configurazioni.

5.3.5 Macchine con Undo (UTM)

Hanno l'operazione UNDO che annulla l'ultima operazione eseguita.

Equivalenza: Una TM standard può simulare le UTM mantenendo una cronologia delle configurazioni precedenti.

5.3.6 Macchine Save-Restore (SRTM)

Hanno operazioni SAVE (salva configurazione corrente) e RESTORE (ripristina configurazione salvata).

Equivalenza: Una TM multinastro può simulare le SRTM usando nastri aggiuntivi per memorizzare le configurazioni salvate.

6 Esempi di dimostrazioni complete

6.1 Esempio 1: FORTY-TWO è indecidibile

$FORTY{-}TWO = \{M \mid M$ termina la computazione su w avendo solo 42 sul nastro$\}$

Dimostrazione: Riduciamo dal problema della fermata. Costruiamo una funzione di riduzione f da H a FORTY-TWO: $f((M, w)) = M'$, dove M' è una TM che:

1. Simula M su input w
2. Se M si ferma, M' cancella il suo nastro e scrive 42
3. Se M non si ferma, M' continua a girare all'infinito

$(M, w) \in H$ se e solo se $M' \in$ FORTY-TWO.

6.2 Esempio 2: LOADBALANCE è NP-completo

$LOADBALANCE = \{(m, T, k) \mid$ esiste un assegnamento A degli n lavori su m linee di produzione tale che $makespan(A) \leq k\}$

Parte 1 - In NP: Verificatore che controlla se un assegnamento proposto ha makespan $\leq k$.

Parte 2 - NP-hard: Riduzione da SETPARTITIONING. Data un'istanza di SETPARTITIONING con insieme S, costruiamo:

- $m = 2$ (due linee di produzione)
- T contiene gli elementi di S come lavori
- $k = \frac{1}{2} \sum_{x \in S} x$

Esiste una partizione bilanciata di S se e solo se esiste un assegnamento con makespan $\leq k$.

6.3 Esempio 3: Equivalenza Tag-Turing TM Standard

Tag-Turing → TM Standard: Costruiamo una TM standard S che simula una tag-Turing machine M:

1. S usa il nastro diviso in configurazioni separate dal simbolo #
2. Per simulare una transizione, S scorre il nastro per trovare la posizione della testina di lettura e determina il simbolo letto
3. Se la funzione di transizione stabilisce che la testina di lettura deve spostarsi a destra, S sposta il pallino nella cella immediatamente a destra
4. Se necessario per stabilire la mossa fare, S sposta la testina e scrive il simbolo stabilito dalla funzione di transizione nella cella e sposta la marcatura

TM Standard → Tag-Turing: Una TM standard è un caso particolare di tag-Turing machine dove la testina di lettura e scrittura coincidono.

7 Schemi risolutivi

7.1 Schema per indecidibilità

1. Identificare il problema noto indecidibile da cui ridurre (spesso A_{TM})
2. Costruire una funzione di riduzione f che trasforma istanze del problema noto in istanze del problema target
3. Dimostrare che f è calcolabile in tempo finito
4. Dimostrare che $x \in$ Problema noto $\Leftrightarrow f(x) \in$ Problema target
5. Concludere per contraddizione

7.2 Schema per NP-completezza

1. **Parte NP:** Costruire un verificatore polinomiale
   - Definire il formato del certificato
   - Specificare le verifiche da effettuare
   - Dimostrare che le verifiche richiedono tempo polinomiale
2. **Parte NP-hard:** Riduzione da un problema NP-hard noto
   - Scegliere il problema sorgente appropriato
   - Costruire la funzione di riduzione
   - Dimostrare la correttezza della riduzione
   - Verificare che la riduzione sia polinomiale

7.3 Schema per equivalenza tra varianti TM

1. **Direzione 1:** Variante $\rightarrow$ TM Standard
   - Descrivere come simulare ogni operazione della variante
   - Dimostrare che la simulazione termina se e solo se la variante termina
   - Dimostrare che la simulazione accetta se e solo se la variante accetta
2. **Direzione 2:** TM Standard $\rightarrow$ Variante
   - Spesso la TM standard è un caso particolare della variante
   - Altrimenti, descrivere la codifica delle operazioni standard nella variante