Automi e Linguaggi Formali
Università degli Studi di Padova
Maggio 2024

Homework 7 - Decidibilità ed indecidibilità

Gabriel Rovesti

1. Sia $INFINITE_{PDA} = \{\langle M \rangle \mid A$ è un PDA ed $L(M)$ è un linguaggio infinito. Si mostri che $INFINITE_{PDA}$ è decidibile.

   Soluzione

   Costruiamo una Turing Machine TM in grado di decidere il problema. Dato il PDA M, possiamo convertirlo in una CFG equivalente G' in forma normale di Chomsky e controllare se G' possiede una derivazione $A \to ua$ dove $u, v \in \Sigma$.

   Se $A \to uAv$ è una derivazione in G', allora $L(M)$ è un linguaggio infinito, altrimenti L(M) è finito. La seguente TM M decide $INFINITE_{PDA}$:

   $M =$ "Su input $\langle M \rangle$, dove M è un PDA:

   (a) Converte M in una grammatica CFG G

   (b) Converte G in forma normale di Chomsky, diventando G'

   (c) Se G' include la derivazione $A \to uAv$ accetta, altrimenti rifiuta.

2. Diciamo che una variabile A nella CFG G è utilizzabile se compare in qualche derivazione di qualche stringa $w \in G$. Data una CFG G e una variabile A, consideriamo il problema di verificare se A è *utilizzabile*. Formulare questo problema come un linguaggio e dimostrare che è decidibile.

   Soluzione

   Per dimostrare che L è decidibile, progetteremo una TM che deciderà L. Chiameremo questa TM M. Ecco una descrizione di alto livello di M:

   (a) Input: $\langle$G, A$\rangle$, dove G è un CFG e A è una variabile di G

   (b) Per ogni simbolo terminale t in G:

   - Tentare di derivare t da A, registrando ogni passo di derivazione e la stringa risultante

- Se la derivazione ha successo, accettare
- Se la derivazione non ha successo, continuare con il simbolo terminale successivo

(c) Se nessuna derivazione ha successo, rifiutare

La vera e propria macchina di Turing M funzionerà come segue:

(a) Inizializzare l'insieme S come $\{A\}$

(b) Se nell'insieme S c'è un simbolo non terminale B che non è stato elaborato:

- Contrassegnare B come elaborato
- Per ogni regola di produzione in G con B sul lato sinistro:
  i. Applicare la regola di produzione alla stringa contenente B
  ii. Aggiungere la stringa risultante all'insieme S
  iii. Se la stringa finale contiene solo simboli terminali e B è stato sostituito da simboli terminali, accettare

(c) Se nessuna derivazione ha successo, rifiutare

Questa TM M, dopo aver ricevuto una coppia di input $, A\rangle$, cercherà la derivazione di una stringa terminale da A applicando le regole di produzione di G in modo ricorsivo fino a quando non avrà derivato una stringa terminale da A o avrà esaurito tutte le derivazioni possibili. In entrambi i casi, M si ferma. Se M trova una derivazione, accetta e si dimostra che L è decidibile.

3. Uno *stato inutile* in un automa a pila non viene mai inserito in nessuna stringa di input. Consideriamo il problema di determinare se un automa pushdown ha degli stati inutili. Formulate questo problema come un linguaggio e dimostrate che è decidibile.

   Soluzione

   Sia $U_{PDA} = \{\text{P} \mid P \text{ il PDA che ha gli stati inutili}\}$. Creiamo una TM T che decide U_{PDA}. Per prima cosa, definiamo $E_{PDA} = \{\text{P} \mid P \text{ è un PDA } L(P) = \emptyset\}$. Mostriamo ora come E_{PDA} sia decidibile con un decisore D:
   $D =$ "Su input P dove P è un PDA:

   1. Convertire P in una CFG equivalente G.
   2. Eseguire E_{CFG} su input G. Output quello che E_{CFG} dà in uscita."

Questo linguaggio è decidibile perché tutti i passi della sua costruzione richiedono un tempo finito e E_{CFG} è un decisore.
$T =$ "Su input P dove $P = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ è un PDA:

1. Per tutti $\forall q \in Q$:
   a. Modificare P affinché $F = \{q\}$.
   b. Eseguire E_{PDA} su input P. If E_{PDA} accetta, *accept* P.
2. $Rifiuta$ P."

Dato che E_{PDA} è decidibile, U_{PDA} è decidibile.

4. Sia $S = \{\langle M \rangle \mid M$ è un DFA che accetta w^R in qualsiasi caso accetti $w\}$. Si mostri che S è decidibile.

   Soluzione

   Per dimostrare che S è decidibile, costruiamo un decisore D per S come segue (sia C una TM che decide l'EQ_{DFA}):

   $D =$ Su input $\langle M \rangle$:

   (a) Costruisci un NFA M' tale che $\langle M' \rangle = \{w^R \mid w \in L(M)\}$
   (b) Convertire M' in un DFA M'' equivalente
   (c) Usa C per confrontare $L(M'')$ ed $L(M)$
   (d) Se $L(M'') = L(M)$, accetta, altrimenti rifiuta

   Nella TM di cui sopra, il passo 1 può essere eseguito convertendo M in M' in passi finiti. L'idea è di (i) invertire le direzioni di tutte le frecce di (ii) creare un nuovo stato q' in M' e collegare q' a ogni stato finale originale di M con ϵ-transizioni, e (iii) rendere lo stato iniziale originale di M uno stato finale di M'. È facile verificare che $\langle M' \rangle = \{w^R \mid w \in L(M)\}$ Inoltre, sia il passo B che il passo C possono essere eseguiti in passi finiti. Quindi, D si svolge in passi finiti è quindi un decisore.

5. Si consideri il problema di determinare se una macchina di Turing M su un input w tenta mai di spostare la sua testina a sinistra quando la sua testina si trova sulla cella più a sinistra del nastro. Formulate questo problema come un linguaggio e dimostrate che è indecidibile.

Soluzione

Definiamo il linguaggio L come:

$L = \{\langle M, w\rangle \mid M$ è una macchina di Turing che tenta di spostare la sua testina a sinistra quando si trova sulla cella più a sinistra del nastro su input $w\}$

Per dimostrare che L è indecidibile, utilizzeremo una riduzione dal problema della macchina di Turing sull'input vuoto, ovvero $A_{TM} = \{\langle M\rangle \mid M$ si arresta sull'input vuoto$\}$, che sappiamo essere indecidibile.

Supponiamo per assurdo che esista una macchina di Turing R che decide L. Costruiamo una macchina di Turing S che decide A_{TM} utilizzando R come subroutine:

Algorithm 1 Macchina di Turing S che decide A_{TM}

Inputinput $\langle M\rangle$, una codifica di una macchina di Turing M Costruisci una nuova macchina di Turing M' che simula M sull'input vuoto con la seguente modifica:
- Se M si arresta, M' tenta di spostare la sua testina a sinistra quando si trova sulla cella più a sinistra del nastro.
- Se M non si arresta, M' entra in un loop infinito senza mai tentare di spostare la sua testina a sinistra quando si trova sulla cella più a sinistra del nastro.
Esegui R su input $\langle M', \varepsilon\rangle$, dove ε rappresenta la stringa vuota.
R accetta **return** M si arresta sull'input vuoto **return** M non si arresta sull'input vuoto

Se M si arresta sull'input vuoto, allora M' tenterà di spostare la sua testina a sinistra quando si trova sulla cella più a sinistra del nastro. Quindi, $\langle M', \varepsilon\rangle \in L$, e R accetterà.

Se M non si arresta sull'input vuoto, allora M' non tenterà mai di spostare la sua testina a sinistra quando si trova sulla cella più a sinistra del nastro. Quindi, $\langle M', \varepsilon\rangle \notin L$, e R rifiuterà.

Ciò implica che S decide correttamente A_{TM}. Tuttavia, sappiamo che A_{TM} è indecidibile, il che porta a una contraddizione. Pertanto, la nostra ipotesi che L sia decidibile deve essere falsa, e concludiamo che L è indecidibile.

6. Si consideri il problema di determinare se una macchina di Turing a nastro singolo scrive mai un simbolo vuoto su un simbolo non vuoto

durante il corso della sua computazione su una qualsiasi stringa di input. Formulate questo problema come un linguaggio e dimostrate che è indecidibile.

Soluzione

Definiamo il linguaggio L come l'insieme di tutte le descrizioni delle macchine di Turing a nastro singolo (codificate in modo appropriato) che scrivono almeno una volta un simbolo vuoto su un simbolo non vuoto durante il corso della loro computazione su qualsiasi stringa di input.

Formalmente: $L = \{\langle M \rangle \mid$ M è una macchina di Turing a nastro singolo che scrive almeno una volta un simbolo vuoto su un simbolo non vuoto durante il corso della sua computazione su qualsiasi stringa di input$\}$

Per dimostrare che il linguaggio L è indecidibile, useremo una riduzione dalla tecnica di diagonalizzazione di Cantor utilizzata per dimostrare l'indecidibilità del problema dell'arresto (halting problem). Questa viene fatta per dimostrarvi come usare questo ragionamento, utile in generale perlomeno a livello logico.

Supponiamo, per assurdo, che esista un decisore D che decide L. Costruiremo una nuova macchina di Turing M che contraddice l'esistenza di D.

La macchina di Turing M funziona come segue:

- Sulla stringa di input w, M simula D su $\langle M \rangle$ (la codifica di sé stessa).
- Se D accetta $\langle M \rangle$ (cioè, se D determina che M scrive almeno una volta un simbolo vuoto su un simbolo non vuoto), allora M entra in un loop infinito senza scrivere mai un simbolo vuoto su un simbolo non vuoto.
- Se D rifiuta $\langle M \rangle$ (cioè, se D determina che M non scrive mai un simbolo vuoto su un simbolo non vuoto), allora M scrive un simbolo vuoto su un simbolo non vuoto e si ferma.

Ora, consideriamo il comportamento di M:

- Se M appartenesse a L, allora D dovrebbe accettare $\langle M \rangle$, ma in questo caso, M entrerebbe in un loop infinito senza scrivere mai un simbolo vuoto su un simbolo non vuoto, contraddizione.

- Se M non appartenesse a L, allora D dovrebbe rifiutare $\langle M \rangle$, ma in questo caso, M scriverebbe un simbolo vuoto su un simbolo non vuoto, contraddizione.

Quindi, in entrambi i casi, abbiamo una contraddizione con il supposto comportamento di D. In questo modo, il problema può dirsi indecidibile.