ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Departement Informatik

Open Class – Sieben Wunder der Informatik

Prof. Dr. Juraj Hromkovič

Lösungsvorschläge für die Übungsaufgaben – Blatt 3

Zürich, 30. November 2005

Lösung zu Aufgabe 8

Wir wollen zeigen, dass das Entscheidungsproblem $(\mathbb{N}, \text{2DIAG})$ algorithmisch unentscheidbar ist. Hierbei bezeichne 2DIAG die Menge aller geraden natürlichen Zahlen $2i$, so dass $2i$ nicht in $L(P_i)$ enthalten ist. Wir verwenden die Bezeichnung $L_2(P_i)$ für die Menge aller geraden Zahlen in $L(P_i)$.
Wir verwenden für unseren Beweis die Diagonalisierungsmethode: Sei P_i das i-te Programm in der Reihenfolge, wie sie in der Vorlesung angegeben wurde. Dann können wir die Mengen $L_2(P_i)$ in einer unendlichen Tabelle darstellen, wie in Abbildung 1 gezeigt.

	0	2	4	6	8	$\cdots$	$2i$	$\cdots$
$L_2(P_0)$	**0**	1	1	0	1	$\cdots$		$\cdots$
$L_2(P_1)$	1	**1**	0	0	0	$\cdots$		$\cdots$
$L_2(P_2)$	1	0	**1**	0	1	$\cdots$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$L_2(P_i)$	0	0	1	0	0	$\cdots$	**$a_{i,2i}$**	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	
2DIAG	1	0	0		$\cdots$		$\overline{a_{i,2i}}$	$\cdots$

Abbildung 1: Die Diagonalisierungsmethode für 2DIAG.
Es gilt $\overline{a_{i,2i}} = 1$, falls $a_{i,2i} = 0$, und $\overline{a_{i,2i}} = 0$, falls $a_{i,2i} = 1$.

Also kann man die Menge 2DIAG bilden, indem man die Zahl $2i$ genau dann in 2DIAG aufnimmt, wenn das Feld an der Kreuzung der i-ten Zeile mit der Spalte $2i$ eine 0 enthält, wenn also $2i$ nicht in $L_2(P_i)$ enthalten ist und somit auch nicht in $L(P_i)$.
Damit unterscheidet sich 2DIAG von $L_2(P_i)$ mindestens in der Spalte $2i$. Also unterscheidet sich 2DIAG von allen Zeilen der Tabelle, kann also von keinem der gelisteten Programme erkannt werden.

Lösung zu Aufgabe 9

Ohne weiteres darf man nicht schliessen, dass DIAG_2 unentscheidbar ist. Denn die Argumente aus der Diagonalisierungsmethode lassen hier nur den Schluss zu, dass kein Programm mit einer geraden Nummer (bzgl. der in der Veranstaltung eingeführten Programmnummerierung) in der Lage ist, DIAG_2 zu entscheiden. Es verbleiben unendlich viele weitere Programme, nämlich all jene mit ungeraden Nummern, über die wir – zumindest auf diese Art und Weise – keine Aussage treffen können.

Lösung zu Aufgabe 10

Wir wollen zeigen, dass das universelle Problem UNIV nicht algorithmisch entscheidbar ist. Hierfür verwenden wir eine Reduktion von DIAG auf UNIV. Diese Reduktion ist in Abbildung 2 schematisch dargestellt.

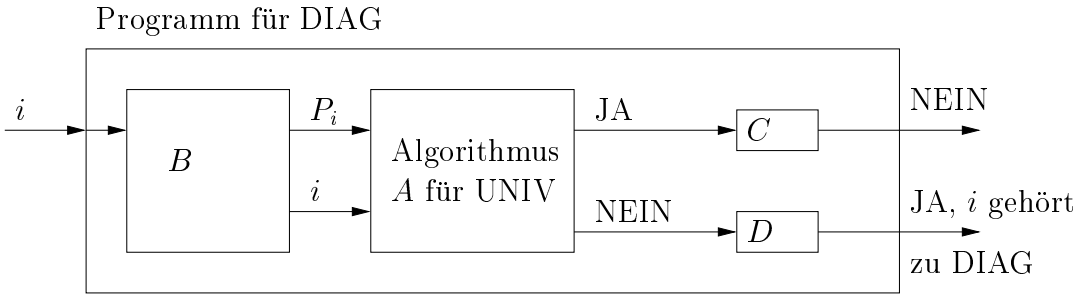

Abbildung 2: Die Reduktion von DIAG auf UNIV.

Das Programm B berechnet für das gegebene i eine Darstellung von P_i und übermittelt i und P_i an den hypothetischen Algorithmus A für UNIV. Falls A bestimmt, dass i in $L(P_i)$ liegt (also „JA“ ausgibt), dann ist i nicht in DIAG enthalten, und das Programm C wandelt die Ausgabe in „NEIN“ um. Falls A bestimmt, dass i nicht in $L(P_i)$ liegt (also „NEIN“ ausgibt), dann ist i in DIAG enthalten, und das Programm D wandelt die Ausgabe in „JA“ um.
Einen Algorithmus für UNIV könnte man also dazu verwenden, einen Algorithmus für DIAG zu konstruieren. Da DIAG aber unentscheidbar ist, folgt hieraus, dass auch UNIV unentscheidbar sein muss.

Lösung zu Bonus-Aufgabe 3

Wir wollen zeigen, dass das Problem LEER nicht algorithmisch entscheidbar ist. Hierfür verwenden wir eine Reduktion von UNIV auf LEER. Diese Reduktion ist in Abbildung 3 schematisch dargestellt.

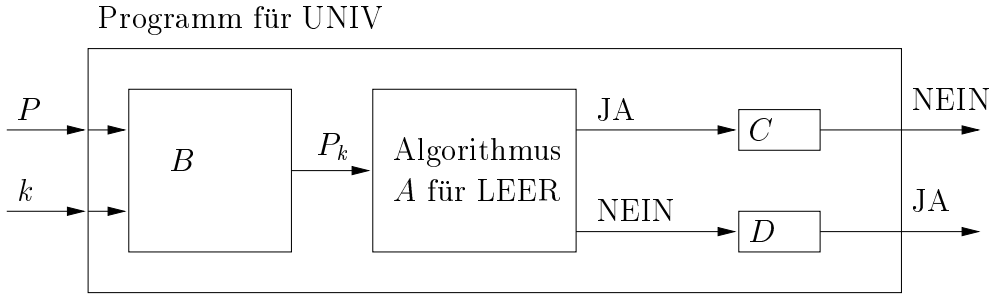

Abbildung 3: Die Reduktion von UNIV auf LEER.

Das Programm B berechnet aus seinen beiden Eingaben, dem Programmtext P und der Zahl k, einen neuen Programmtext P_k und übermittelt P_k an den hypothetischen Algorithmus A für LEER. Hierbei geht P_k aus P hervor, indem jeder Befehl der Form

$$\texttt{Lese ein in Register}(n)$$

in P durch den Befehl

$$\texttt{Register}(n) \leftarrow k$$

ausgetauscht wird. Da wir annehmen (dürfen), dass P genau einmal in ein Register von der Eingabe hineinliest, wird an genau dieser einen Stelle in der Ausführung von P_k stattdessen die Konstante k in dasjenige Register hineingeschrieben werden. (Nota bene: Weder P noch P_k werden zu irgendeinem Zeitpunkt „aufgerufen“ oder „simuliert“. Die Arbeit des Algorithmus B besteht in einer rein syntaktischen Modifikation eines Programmtextes.)
Es ist leicht einzusehen, dass P_k stets dieselbe Arbeit verrichtet, die P verrichten würde, wenn man es mit der Eingabe k laufen liesse. Insbesondere ist völlig unerheblich, welche Eingabe P_k erhält, denn da P_k keinen Befehl der Form

$$\texttt{Lese ein in Register}(n)$$

enthält, wird die Eingabe gänzlich ignoriert.
Da P_k also unabhängig von der eigenen Eingabe stets genauso rechnet wie das Programm P auf der Eingabe k, folgt, dass P_k auch dasselbe Verhalten an den Tag legt und insbesondere genau in dem Fall nach endlicher Zeit „JA“ ausgibt, für den auch P (auf der Eingabe k) das tut. Also ist P_k genau dann in LEER enthalten, wenn P auf der Eingabe k entweder unendlich lange läuft oder aber nicht „JA“ ausgibt.
Dieses Wissen nutzt unsere Reduktion nun, um UNIV zu entscheiden: Wenn P_k in LEER liegt, so antwortet der Algorithmus A mit „NEIN“. Liegt P_k jedoch nicht in LEER, so antwortet er mit „JA“.