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

Open Class – Sieben Wunder der Informatik

Prof. Dr. Juraj Hromkovič

Übungsaufgaben – Blatt 3

Zürich, 23. November 2006

Zusammenfassung und Aufgaben

Ein *Entscheidungsproblem* besteht darin, zu entscheiden, ob ein gegebenes Objekt eine gewisse Eigenschaft hat. Wir beginnen mit einfachen Objekten, den natürlichen Zahlen.
Wir betrachten das Erkennnen einer beliebigen Teilmenge von $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ als Entscheidungsproblem. Mit jeder Teilmenge L von $\mathbb{N}$ assoziieren wir die darin bestehende Aufgabe, zu entscheiden, ob eine gegebene natürliche Zahl i in L enthalten ist. Wir sagen, dass ein Algorithmus A *eine Menge* L *erkennt* oder *das Entscheidungsproblem* $(\mathbb{N}, L)$ *löst,* wenn A für jede Eingabe $n \in \mathbb{N}$

(i) die Ausgabe „JA“ (oder „1“) liefert, falls n in L enthalten ist, und

(ii) die Ausgabe „NEIN“ (oder „0“) liefert, falls n nicht in L enthalten ist.

Wenn A auf der Eingabe n mit „JA“ endet, dann sagen wir auch, dass A die Zahl n *akzeptiert.*
Zum Beispiel können wir die Menge PRIM, die alle Primzahlen $2, 3, 5, 7, 11, 13, \ldots$ enthält, mit einem geeigneten Algorithmus erkennen, der einfach die Teilbarkeit der gegebenen Zahl n überprüft.
Ein Entscheidungsproblem heisst *algorithmisch entscheidbar*, wenn ein Algorithmus existiert, der das Entscheidungsproblem löst. Ein Entscheidungsproblem heisst *algorithmisch unentscheidbar* (oder *unlösbar*), wenn es keinen Algorithmus gibt, der das Problem löst.
Wir haben gezeigt, dass es algorithmisch unlösbare Probleme gibt. Das erste Beispiel dieser Art ist das Entscheidungsproblem $(\mathbb{N}, \text{DIAG})$. Die Menge DIAG kann man wie folgt konstruieren:

1. Zuerst nummeriert man alle Programme, die als Eingabe eine natürliche Zahl erhalten und als Ausgabe „JA“ oder „NEIN“ liefern. Diese Nummerierung erhält man, indem die längeren hinter den kürzeren Programmen einsortiert werden und gleich lange Programme ihren Buchstaben nach wie im Wörterbuch sortiert werden. Dabei sei P_i die Bezeichnung des i-ten Programms in der so sortierten Folge.

2. Jedes Programm P_i definiert eine Menge $L(P_i)$ wie folgt: $L(P_i)$ enthält alle natürlichen Zahlen, für die P_i seine Arbeit mit der Ausgabe „JA“ beendet. Alle Zahlen, für die P_i „NEIN“ ausgibt oder auf denen P_i unendlich lange rechnet, sind nicht in $L(P_i)$ enthalten. Jede Menge $L(P_i)$ kann als eine unendliche Folge von Nullen und Einsen dargestellt werden, in der an der j-ten Stelle genau dann eine Eins steht, wenn j in $L(P_i)$ enthalten ist.

3. Wir definieren DIAG als die Menge aller natürlichen Zahlen i, so dass i nicht in $L(P_i)$ enthalten ist.

Die Konstruktion von DIAG nennt man die *Diagonalisierungsmethode*. Sie kann anschaulich wie in der folgenden Abbildung 1 dargestellt werden:

	0	1	2	3	4	$\cdots$	i	$\cdots$
$L(P_0)$	**0**	1	1	0	1	$\cdots$		$\cdots$
$L(P_1)$	1	**1**	0	0	0	$\cdots$		$\cdots$
$L(P_2)$	1	0	**1**	0	1	$\cdots$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$L(P_i)$	0	0	1	0	0	$\cdots$	**a_{ii}**	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	
DIAG	1	0	0	$\cdots$			$\overline{a_{ii}}$	$\cdots$

Abbildung 1: Die Diagonalisierungsmethode.
Es gilt $\overline{a_{ii}} = 1$, falls $a_{ii} = 0$, und $\overline{a_{ii}} = 0$, falls $a_{ii} = 1$.

Die zweidimensionale unendliche Tabelle besteht aus unendlichen Zeilen, wobei die i-te Zeile die Darstellung der Menge $L(P_i)$ als unendliche Folge von Nullen und Einsen ist. Die Spalten der Tabelle sind mit den natürlichen Zahlen $0, 1, 2, \ldots$ nummeriert. Das Feld an der Kreuzung der k-ten Zeile mit der j-ten Spalte enthält eine 1, wenn die Zahl j in der k-ten Menge $L(P_k)$ enthalten ist, und eine 0 sonst. Nun kann man DIAG bilden, indem man i in DIAG genau dann aufnimmt, wenn das Feld an der Kreuzung der i-ten Zeile mit

der i-ten Spalte (also auf der Diagonale der Tabelle) eine 0 enthält, wenn also i *nicht* in $L(P_i)$ enthalten ist.
Somit unterscheidet sich DIAG von $L(P_i)$ mindestens in der Zugehörigkeit der Zahl i. Also unterscheidet sich DIAG von allen Zeilen der Tabelle, kann also von keinem der aufgelisteten Programme erkannt werden.

Aufgabe 8

Wir definieren 2DIAG als die Menge aller geraden natürlichen Zahlen $2i$, so dass $2i$ nicht in $L(P_i)$ enthalten ist.
Ist das Entscheidungsproblem $(\mathbb{N}, \text{2DIAG})$ algorithmisch entscheidbar oder nicht? Begründen Sie Ihre Antwort und zeichnen Sie dazu auch ein Bild analog zu Abbildung 1.
(Bemerkung: Wir betrachten $0 = 2 \cdot 0$ als eine gerade Zahl.) **10 Punkte**

Aufgabe 9

Wir definieren DIAG_2 als die Menge aller geraden natürlichen Zahlen $2i$, so dass $2i$ nicht in $L(P_{2i})$ enthalten ist. Was kann man über DIAG_2 aussagen? **10 Punkte**

Um von solchen abstrakten Diagonalmengen zu konkreten Problemen zu kommen, haben wir den Begriff der *Reduktion* eingeführt.
Ein Problem P_1 ist auf ein Problem P_2 *reduzierbar,* wenn die Existenz eines Algorithmus für P_2 die Existenz eines Algorithmus für P_1 impliziert.

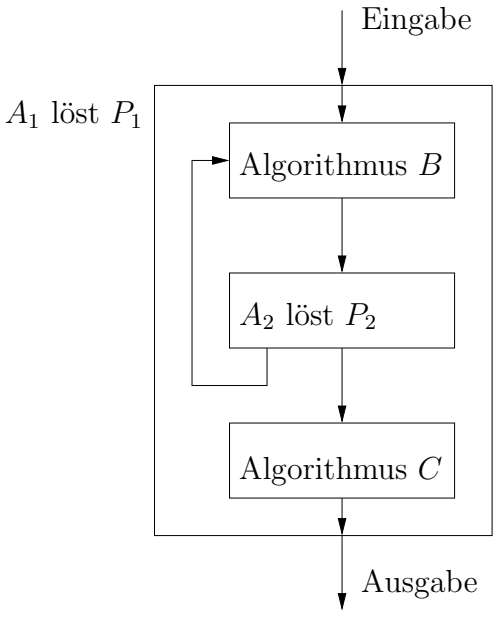

Abbildung 2: Schematische Darstellung einer Reduktion

In der Abbildung 2 hat man hypothetisch einen Algorithmus A_2 für P_2 gegeben. Diesen Algorithmus nutzt man als Teilprogramm in einem Algorithmus A_1, um P_1 zu lösen.
Wenn P_1 auf P_2 reduzierbar ist, dann sagen wir, dass P_1 *algorithmisch nicht schwerer als* P_2 ist, oder dass P_2 *algorithmisch nicht leichter als* P_1 ist, weil die algorithmische Lösbarkeit von P_2 die algorithmische Lösbarkeit von P_1 garantiert.
Der Begriff der Reduktion ermöglicht es uns, Methoden zum Beweis der algorithmischen Unlösbarkeit konkreter Probleme zu entwickeln. Wenn man zum Beispiel zeigen kann, dass aus der algorithmischen Lösbarkeit eines Problems P die algorithmische Lösbarkeit von $(\mathbb{N}, \text{DIAG})$ folgt, dann kann man daraus schliessen, dass P algorithmisch unlösbar ist.
Wir sind diesen Weg für das *Halteproblem* gegangen.

Halteproblem: HALT

Eingabe: Ein Programm P und eine natürliche Zahl k.

Ausgabe: „JA“, falls P auf der Eingabe k hält, also nur endlich lange arbeitet, „NEIN“, falls P auf k unendlich lange arbeitet.

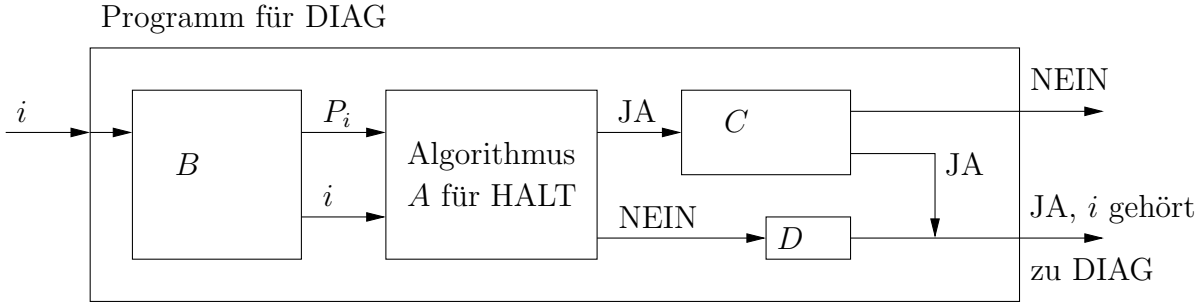

Abbildung 3: Die Reduktion von DIAG auf HALT.

Die Abbildung 3 zeigt die Reduktion von DIAG auf HALT.
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 HALT. Falls A bestimmt, dass P_i unendlich lange auf i arbeitet (also die Ausgabe „NEIN“ liefert), dann ist i nicht in $L(P_i)$ und somit ist i in DIAG. Die Antwort „NEIN“ von Algorithmus A wird deshalb von dem Programm D in die Antwort „JA“ umgewandelt. Falls A bestimmt, dass P_i auf i hält (also „JA“ ausgibt), dann können wir mit dem Programm C die Arbeit von P_i auf i in endlicher Zeit simulieren. Falls P_i die Eingabe i akzeptiert, dann ist i nicht in DIAG (weil i in $L(P_i)$ ist) und C gibt die Antwort „NEIN“ aus. Falls P_i die Zahl i nicht akzeptiert, dann ist i in DIAG, und somit gibt C die Antwort „JA“ aus.

Aufgabe 10

Wir betrachten das sogenannte *universelle Problem:*

Universelles Problem: UNIV

Eingabe: Ein Programm P und eine natürliche Zahl k.

Ausgabe: „JA“, falls k in $L(P)$ liegt, und „NEIN“, falls k nicht in $L(P)$ liegt.

Zeigen Sie mittels der Reduktionsmethode, dass UNIV nicht algorithmisch lösbar ist.

10 Punkte

Bonus-Aufgabe 3

Zeigen Sie, dass das folgende *Leerheitsproblem* nicht algorithmisch lösbar ist:

Leerheitsproblem: LEER

Eingabe: Ein Programm A.

Ausgabe: „JA“, falls $L(A) = \emptyset$, A also keine Zahl akzeptiert, und „NEIN“ sonst.

10 Bonus-Punkte

Ihre Lösungen zu den Aufgaben können Sie entweder persönlich bei der Open-Class-Veranstaltung am 23. November 2005 abgeben oder bis zum 23. November per E-Mail (möglichst als PDF-Datei) an `hjb@inf.ethz.ch` oder per Post an folgende Adresse schicken:

Dr. Hans-Joachim Böckenhauer
Informationstechnologie und Ausbildung
ETH Zentrum CAB F 11.1
Universitätsstrasse 6
8092 Zürich

Bitte vergessen Sie nicht, Ihre Lösung mit Ihrem Namen und Ihrer E-Mail-Adresse zu versehen.
Falls Ihre Lösung uns bis zum 21. November 2005 erreicht, können Sie die korrigierte Lösung bereits in der Veranstaltung am 23. November abholen (sonst eine Woche später).