Universität Siegen
Lehrstuhl Theoretische Informatik
Markus Lohrey

Musterlösung zu Übungsblatt 14

Aufgabe 1. Die Softwarefirma HALTING & CO. KG bietet folgende Produkte zur Programmverifikation an.

(a) Produkt A überprüft, ob ein gegebenes Programm auf allen Eingaben terminiert.

(b) Produkt B überprüft, ob ein gegebenes Programm auf einer gegebenen Eingabe höchstens 1,000 Rechenschritte durchführt.

(c) Produkt C überprüft, ob ein gegebenes Programm auf einer gegebenen Eingabe höchstens 1 GB Speicher benötigt.

(d) Produkt D überprüft, ob ein gegebenes Programm niemals die Ausgabe 123 produziert.

Welche Produkte kann es tatsächlich geben?

Lösung zu Aufgabe 1.

(a) Falsch. Das spezielle Halteproblem (Folie 477) kann auf das Problem a) reduziert werden. Da das spezielle Halteproblem unentscheidbar ist, folgt dass auch dieses Problem unentscheidbar ist.

Die Reduktion funktioniert wie folgt: Wir zeigen, dass wenn man prüfen könnte ob ein gegebenes Programm (eine Turing-Maschine) auf allen Eingaben terminiert, so könnte man auch das spezielle Halteproblem entscheiden (welches prüft ob eine Turing-Maschine M_w auf der eigenen Kodierung w terminiert). Dazu modifizieren wir eine gegebene Turing-Maschine so, dass zu Beginn die eigentliche Eingabe (egal welche) gegen die Kodierung der gegebenen Turing-Maschine ausgetauscht wird, dass heißt die Turing-Maschine überschreibt den initialen Bandinhalt immer durch die entsprechende Kodierung der Turing-Maschine. Anschließend arbeitet die Turing-Maschine genau wie originale Turing-Maschine. Diese modifizierte Turing-Maschine terminiert nun auf allen Eingaben genau dann, wenn die originale Turing-Maschine auf der eigenen Kodierung terminiert. Somit könnte man das spezielle Halteproblem mit Hilfe von a) entscheiden, was zur Folge hat, dass a) unentscheidbar ist.

(b) Wahr. Das Programm wird unter der gegebenen Eingabe simuliert und die Rechenschritte mitgezählt. Falls das simulierte Programm bis zum 1000. Rechenschritt terminiert, meldet der Simulator «terminiert». Andernfalls wird die Simulation abgebrochen und der Simulator meldet «terminiert nicht bei bis zu 1000 Rechenschritten» .

(c) Wahr. Das Programm wird unter der gegebenen Eingabe simuliert und alle bereits erreichten Konfigurationen gespeichert. Wichtig ist, dass es mit 1 GB Speicher nur endlich viele Konfigurationen gibt, die in diesem begrenzten Speicher arbeiten. Sollte eine Konfiguration auftauchen, die mehr als 1 GB Speicher benötigt, bricht der Simulator mit «braucht mehr als 1GB Speicher» ab. Sollte das simulierte Programm terminieren ohne jemals eine zu große Konfiguration zu erreichen, meldet der Simulator «das Programm terminiert». Der letzte mögliche Fall ist der, dass eine bereits erreichte Konfiguration mit weniger als 1GB erneut erreicht wird (da es nur endlich viele Konfigurationen dieser Art gibt) ohne dass zuvor terminiert wurde oder eine zu große Konfiguration besucht wurde. In diesem Fall würde sich das Programm also in eine Endlosschleife begeben und der Simulator bricht mit «das Programm terminiert nicht» ab.

(d) Falsch. Zuerst zeigen wir, dass es unentscheidbar ist, ob eine Turing-Maschine auf keiner Eingabe terminiert. Dazu zeigen wir eine Reduktion auf das Komplement des speziellen Halteproblem (das Komplement einer unentscheidbaren Sprache ist wiederum unentscheidbar), d.h., wenn man prüfen könnte, ob eine Turing-Maschine auf keiner Eingabe terminiert, so könnte man auch prüfen ob eine Turing-Maschine nicht auf seiner eigenen Kodierung terminiert. Dazu modifizieren wir die gegebene Turing-Maschine genau wie bei a), d.h. eine beliebige Eingabe wird zu Beginn durch die Kodierung der gegebenen Turing-Maschine überschrieben und anschließend arbeitet die modifizierte Turing-Maschine genau wie die ursprüngliche Turing-Maschine. Nun gilt, dass die modifizierte Turing-Maschine auf keiner Eingabe terminiert, genau dann, wenn die originale Turing-Maschine auf der eigenen Kodierung nicht terminiert. Somit könnte man das Komplement des speziellen Halteproblems entscheiden, falls man entscheiden könnte ob eine Turing-Maschine auf keiner Eingabe terminiert, was zur Folge hat, dass dieses Problem ebenfalls unentscheidbar ist.
Im zweiten Schritt geben wir eine Reduktion von d) auf das eben betrachtete unentscheidbare Problem an, welches prüft ob ein Programm auf keiner Eingabe terminiert. Die Reduktion funktioniert wie folgt: Wir zeigen, dass wenn man prüfen könnte ob ein gegebenes Programm niemals die

Ausgabe 123 produziert, so könnte man auch prüfen ob ein Programm auf keiner Eingabe terminiert. Dazu modifizieren wir das gegebene Programm so, dass wann immer das Programm terminiert (also im Falle einer Turing-Maschine in einen Endzustand übergeht), so wird vorher noch die Ausgabe 123 produziert. Dieses modifizierte Programm produziert nun niemals die Ausgabe 123 genau dann, wenn das ursprüngliche Programm auf keiner Eingabe terminiert (da im Umkehrschluss die Ausgabe 123 immer produziert würde, wenn das Programm terminieren würde). Somit könnte man mit Hilfe von d) das unentscheidbare Problem entscheiden, was bedeutet, dass auch d) unentscheidbar ist.

Aufgabe 2. Zeigen Sie, dass die Menge der Primzahlen und die Menge der Quadratzahlen entscheidbar ist.

Lösung zu Aufgabe 2. Dies lässt sich lösen, in dem man beispielsweise ein LOOP-Programm angibt, welches die Ausgabe 1 oder 0 erzeugt, je nachdem, ob die Eingabe (x_1) eine Primzahl (beziehungsweise eine Quadratzahl) ist.[1]

Quadratzahl:

$x_2 := x_1;$
$x_1 := 0;$
$x_3 := 0;$
LOOP x_2 DO
  $x_4 := x_3 \cdot x_3;$
  $x_3 := x_3 + 1;$
  IF $x_4 = x_2$ THEN $x_1 := 1$ END
END

[1]Es werden einige Konstruktionen wie Mulitplikation, Subtraktion, mod und zwei verschiedene IF Anweisungen benutzt. Diese Funktionen sind LOOP berechenbar. Falls noch nicht in den Übungen oder der Vorlesung geschehen, zeigen Sie gerne selbstständig, dass die genutzten Konstrukte LOOP berechenbar sind.

Primzahl:

$x_2 := x_1;$
$x_3 := x_1 - 2;$
$x_4 := 1;$
$x_1 := 1;$
LOOP x_3 DO
  $x_4 := x_4 + 1;$
  $x_5 := x_2 \mod x_4;$
  IF $x_5 = 0$ THEN $x_1 := 0$ END
END

Aufgabe 3. Ein bekanntes Problem aus der Mathematik ist Hilberts zehntes Problem: Gegeben ein Polynom $p(x_1, \ldots, x_n)$ mit ganzzahligen Koeffizienten in n Variablen ($n \geq 1$ beliebig), existieren $x_1, \ldots, x_n \in \mathbb{Z}$ mit $p(x_1, \ldots, x_n) = 0$? Erst 1970 wurde bewiesen, dass dieses Problem unentscheidbar ist.

(a) Ist Hilberts zehntes Problem semi-entscheidbar?

(b) Ist das Komplement semi-entscheidbar?

Lösung zu Aufgabe 3.

(a) Wahr. Die Menge der möglichen Belegungen für ein Polynom über $\mathbb{Z}$ mit n Variablen ist $\mathbb{Z}^n$, welches eine abzählbar unendliche Menge darstellt.[2] Das bedeutet, dass man alle möglichen Belegungen sukzessive durchprobieren kann. Das Verfahren terminiert, wenn es eine Belegung gibt, die das Polynom zu Null auswertet. Ansonsten läuft das Programm für immer weiter. Dies beschreibt also einen Semi-Entscheidungs-Algorithmus.

(b) Falsch. Aus a) und der Unentscheidbarkeit des Problems folgt, dass b) nicht semi-entscheidbar ist (Folie 465).

Aufgabe 4 ($\star$)**.** Der ebenso geniale wie unberechenbare Wissenschaftler und Superbösewicht Doktor Meta ist unentschlossen. Ein neuer Held ist in Siegen aufgetaucht. Theorie-Man streift durch Siegen und bekämpft das Verbrechen. Doktor Meta weiß nicht, wo Theorie-Man seine Basis hat. Das Rennmotorrad von Theorie-Man, eine *Turing 3000*, hat keine Höchstgeschwindigkeit. Doktor Meta möchte Theorie-Man stellen. Er weiß aber nicht, wie er ihn finden kann.

[2] Der Beweis ist analog zur Konstruktion der Funktion $\langle n_1, n_2, \ldots, n_k \rangle$ auf Folie 412, welches eine Bijektion von $\mathbb{N}^k$ nach $\mathbb{N}$ ist.

Zur Vereinfachung des Problems überlegt er sich eine Abstraktion. Er modelliert den Ort von Theorie-Man als natürliche Zahl auf einem eindimensionalen Zahlenstrahl. Die Zeit diskretisiert er ebenfalls als natürliche Zahl. Dann kann er mit einer Funktion $f : \mathbb{N} \to \mathbb{N}$ den Aufenthaltsort von Theorie-Man beschreiben. Er sucht nun ein System, mit dem er garantiert irgendwann zur richtigen Zeit am richtigen Ort ist.

Gegeben sei eine Funktion $f(t) = v \cdot t + s_0$. Die Parameter $v \in \mathbb{N}$ und $s_0 \in \mathbb{N}$ sind unbekannt, aber fest.

Beschreiben Sie einen Algorithmus, der eine Folge $(s_i)_{i\in\mathbb{N}}$ von natürlichen Zahlen ausgibt, so dass ein $j \in \mathbb{N}$ mit $s_j = f(j)$ existiert (der Algorithmus muss nicht terminieren, er muss nur irgendwann ein korrektes Folgenglied ausgeben).

Lösung zu Aufgabe 4. $\mathbb{N}^2$ ist abzählbar unendlich (siehe Folie 412). Folglich existiert eine Bijektion $g\colon \mathbb{N} \to \mathbb{N}^2$ (die Umkehrfunktion der Funktion c auf Folie 412). Sei $g(j) = (j_1, j_2)$, dann definieren wir $s_j = j_1 \cdot j + j_2$ für alle $j \in \mathbb{N}$.

Nun existiert ein $j \in \mathbb{N}$ mit $s_j = f(j)$ nämlich $j = g^{-1}(v, s_0)$, da in diesem Fall $s_j = v \cdot j + s_0 = f(j)$ gilt. Die Funktion g (und somit auch $j \mapsto s_j$) ist eine totale berechenbare Funktion, also gibt es ein Programm, welches $s_1, s_2, \ldots$ ausgibt und auf Grund der obigen Rechnung gilt auch für beliebige v, s_0, dass ein j existiert so, dass $s_j = f(j)$.