technische universität dortmund

Beate Bollig

Gaetano Geck, Thomas Harweg
David Mezlaf, Christopher Spinrath

fakultät für informatik

Abgabe bis spätestens am Dienstag, 04.07.2017, 10:00 Uhr

- **in die Briefkästen im Durchgangsflur, der die 1. Etage der OH 12 mit dem Erdgeschoss der OH 14 verbindet.**

Ansonsten gelten die Hinweise von Blatt 1.

Aufgabe 10.1 [Simulationen von Programmen: WHILE vs. GOTO] **5 Punkte**

Betrachten Sie das folgende GOTO-Programm P.

```
 1: IF x1 = 0 THEN GOTO 10;
 2: x3 := x1 ∸ x2;
 3: x4 := x2 ∸ x1;
 4: IF x3 = 0 THEN GOTO 7;
 5: x1 := x3;
 6: GOTO 2;
 7: IF x4 = 0 THEN GOTO 9;
 8: GOTO 10;
 9: x1 := x1 ∸ x2;
10: HALT
```

Beachten Sie, dass in P GOTO-Programme der Form $x_k := x_i \dot{-} x_j$ genutzt werden. Ihre Semantik ist analog zu der Subtraktion einer Konstanten von einer Variablen definiert, folgt also der intuitiven Bedeutung. Sie dürfen dieses Konstrukt im Folgenden auch als syntaktischen Zucker in WHILE-Programmen benutzen.

Kurzaufgabe (1 Punkt)

Welche Funktion berechnet das Programm P in Abhängigkeit von den Variablen x_1 und x_2? Erläutern Sie kurz und *abstrahierend* die Funktionsweise des Programms.

Hauptaufgabe (4 Punkte)

a) Konstruieren Sie ein zu dem GOTO-Programm P äquivalentes WHILE-Programm gemäß dem in der Beweisskizze zu Satz 13.1 vorgestellten Verfahren, wobei Sie sich eine Simulation *unbedingter* Sprünge überlegen sollen, die sich an der Simulation *bedingter* Sprünge orientiert. Erklären Sie insbesondere Ihre Simulation der unbedingten Sprünge.

Beachten Sie außerdem, dass x_z kein gültiger Variablenname in einem konkreten WHILE-Programm ist. Wählen Sie deshalb einen geeigneten festen Wert für z. Verändern Sie das Programm P *nicht*, bevor Sie es übersetzen. **(2,5 Punkte)**

b) Betrachten Sie das folgende WHILE-Programm P'.

```
1   x2 := 0;
2   WHILE x1 ≠ 0 DO
3     x3 := x1;
4     WHILE x3 ≠ 0 DO
5       x2 := x2 + 1;
6       x3 := x3 ∸ 1
7     END;
8     x1 := x1 ∸ 2
9   END;
10  x1 := x2
```

Konstruieren Sie ein zu P' äquivalentes GOTO-Programm gemäß dem in der Beweisskizze zu Satz 13.2 vorgestellten Verfahren. Verändern Sie das Programm P' *nicht*, bevor Sie es übersetzen. **(1,5 Punkte)**

Aufgabe 10.2 [Church-Turing-These: Mehrstring-Turingmaschinen] 5 Punkte

Kurzaufgabe (1 Punkt)

Beschreiben Sie kurz die Arbeitsweise einer 2-String-Turingmaschine, die mit dem Bandalphabet $\{0, 1, \$, \triangleright, \sqcup\}$ arbeitet und die Sprache

$$\{w\$v \mid w, v \in \{0,1\}^*,\ \#_0(w) = \#_1(v)\}$$

entscheidet, wobei sie den Zeiger auf dem ersten String, also insbesondere auf der Eingabe, nie nach links bewegt. Begründen Sie auch kurz die Korrektheit Ihrer 2-String-Turingmaschine.

Hauptaufgabe (4 Punkte)

Ziel dieser Aufgabe ist es, zu zeigen, dass 2-String-Turingmaschinen genauso ausdrucksstark sind wie Turingmaschinen (die nur auf einem String arbeiten), die Modelle also äquivalent sind. Der Einfachheit halber betrachten wir hier nur solche 2-String-Turingmaschinen, die nie in den Haltezustand h übergehen, also keine Funktion berechnen.

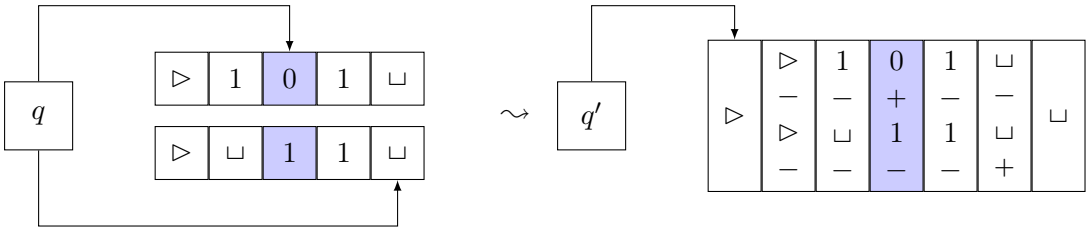
> **Anmerkung:** Das Vorgehen lässt sich außerdem leicht auf Mehrstring-Turingmaschinen erweitern (auch solche, die Funktionen berechnen), was zeigt, dass das Konzept von Mehrstring-Turingmaschinen kein Widerspruch zur Church-Turing-These darstellt.

Da jede Turingmaschine (die keine Funktion berechnet) als eine 2-String-Turingmaschine, die ihren zweiten String nicht benutzt, aufgefasst werden kann, sind 2-String-Turingmaschinen mindestens so ausdruckstark wie Turingmaschinen. Für die Äquivalenz der Modelle bleibt also zu zeigen, dass jede 2-String-Turingmaschine durch eine Turingmaschine simuliert werden kann.

Sei dazu $M = (Q, \Gamma, \delta, s)$ eine 2-String-Turingmaschine. Es gilt, eine 1-String-Turingmaschine $M' = (Q', \Gamma', \delta', s')$ zu konstruieren, die äquivalent zu M ist. Dabei ergeben sich zwei Schwierigkeiten: Erstens müssen beide Strings der 2-String-Turingmaschine M auf dem einzigen String von M' „untergebracht“ werden, und zweitens hat M zwei Zeiger (einen pro String), insbesondere also zwei Zeigerpositionen, während M' nur einen Zeiger hat.

Beide Probleme kann man mit der sogenannten *Spurentechnik* lösen. Die grundlegende Idee ist dabei wie folgt: Das Bandalphabet von M' ist $\Gamma' = (\Gamma \times \{-, +\})^2 \cup \Gamma$. Symbole aus $\Gamma - \{\triangleright, \sqcup\}$ werden nur benötigt, damit M' die gleiche Eingabe und Ausgabe wie M lesen bzw. ausgeben kann.

Während der Simulation von M arbeitet M' also nur mit dem Alphabet $(\Gamma \times \{-,+\})^2 \cup \{\triangleright, \sqcup\}$. Seien σ_1 und σ_2 die zwei Zeichen, die jeweils auf dem ersten bzw. zweiten String von M an der gleichen Position i stehen. Die Turingmaschine M merkt sich diese Information in einem einzigen Zeichen $(\sigma_1, h_1, \sigma_2, h_2)$, wobei h_1 bzw. h_2 genau dann den Wert $+$ annehmen sollen, wenn der Zeiger des ersten bzw. zweiten Strings von M gerade an Position i steht. Diese Repräsentation ist im Folgenden an einem Beispiel illustriert:



In diesem Beispiel steht auf dem ersten String von M an Position $i = 2$ das Zeichen 0 und auf dem zweiten String an Position 2 das Zeichen 1. Außerdem befindet sich der Zeiger des ersten Strings gerade an Position 2, aber der Zeiger des zweiten Strings nicht. Diese Information wird von M' mit dem Zeichen $(0, +, 1, -)$ an Position 3 ihres Strings festgehalten.

a) Die Startkonfiguration von M' ist $(s', (w, 0))$, wobei w die Eingabe – also ein Wort über Γ – ist. Beschreiben Sie, wie M' die Eingabe w durch ein geeignetes Wort w' über $(\Gamma \times \{-,+\})^2 \cup \{\triangleright\}$ ersetzt, um die Startkonfiguration von M zu repräsentieren. Welche Gestalt hat w'? Beachten Sie dabei insbesondere, dass das linke Begrenzungssymbol $\triangleright$ nicht überschrieben werden kann.
(1,5 Punkte)

Ein Berechnungsschritt von M soll nun von M' wie folgt simuliert werden: Sei $q \in Q$ der aktuelle Zustand von M. Ausgehend von dem linken Begrenzungssymbol läuft M' nach rechts über den kompletten String (bis zum ersten Blank $\sqcup$) und merkt sich die zwei Zeichen σ_1 and σ_2, die an den Zeigerpositionen des ersten und zweiten Strings von M stehen. Wir betrachten den Fall, dass

$$\delta(q, (\sigma_1, \sigma_2)) = (p, (\sigma_1', \sigma_2'), (\rightarrow, \leftarrow)) \qquad (\star)$$

gilt. Die Turingmaschine M' läuft dann von rechts nach links (sie ist ja zuvor nach rechts gelaufen) über den String und nimmt die jeweiligen Änderungen vor, um den Berechnungsschritt von M zu simulieren. Entsprechend der Transitionsfunktion von M werden die $+$-Symbole nach links oder rechts „verschoben“ und die Zeichen σ_1 und σ_2 werden überschrieben. Schließlich soll M' zurück zum linken Begrenzungssymbol $\triangleright$ laufen, um danach den nächsten Berechnungsschritt von M simulieren zu können.

b) Formalisieren Sie die oben beschriebene Idee der Spurentechnik, indem Sie Zustände und Transitionen definieren, um einen Berechnungsschritt von M zu simulieren. Beschreiben Sie jeweils den Zweck der Zustände und Transitionen. Gehen Sie davon aus, dass M in dem Berechnungsschritt den Zeiger auf dem ersten String nach rechts und den Zeiger auf dem zweiten String nach links bewegt (vergleiche $(\star)$). Für den aktuellen Zustand von M und die Zeichen an den Zeigerpositionen sind jedoch alle Fälle abzudecken. Sie können weiterhin annehmen, dass der String von M' bereits in der in Aufgabenteil a) illustrierten Gestalt vorliegt, und dass der Zeiger von M' an Position 0 auf dem linken Begrenzungssymbol steht.
(2,5 Punkte)

Aufgabe 10.3 [Berechenbarkeit: Reduktionen] **5 Punkte**

Kurzaufgabe (1 Punkt)

Sei $\Sigma = \{0, 1\}$. Begründen Sie, warum *keine* Reduktion $f : \Sigma^* \to \Sigma^*$ von dem Entscheidungsproblem $L(0(0 + 1)^*)$ auf das Entscheidungsproblem Σ^* existiert. Beachten Sie, dass jede Sprache, also insbesondere jede reguläre Sprache, ein Entscheidungsproblem ist.

Hauptaufgabe (4 Punkte)

Im ersten Teil dieser Aufgabe soll gezeigt werden, dass das folgende Entscheidungsproblem unentscheidbar ist.

Problem: TM-Halt-Even
Gegeben: Eine Turingmaschine M
Frage: Hält M auf allen Eingaben gerader Länge?

Betrachten Sie zu diesem Zweck die Funktion f, die eine Turingmaschine M und ein Eingabewort w auf die Turingmaschine $f(M, w) = M_w$ abbildet. Die Turingmaschine M_w soll wie folgt arbeiten: Sie löscht die Eingabe[1], schreibt dann w unmittelbar hinter das linke Begrenzungssymbol, und verhält sich schließlich genauso wie M auf der Eingabe w.

a) Zeigen Sie, dass f eine Reduktion von TM-Halt auf TM-Halt-Even ist. Gehen Sie dabei insbesondere auf die Berechenbarkeit von f ein. Folgern Sie schließlich die Unentscheidbarkeit von TM-Halt-Even mit Hilfe von f und Resultaten aus der Vorlesung. **(2 Punkte)**

Im zweiten Teil soll ein Entscheidbarkeitsresultat gezeigt werden. Gegeben seien die folgenden Entscheidungsprobleme:

Problem: PDA-Config-Reach
Gegeben: Ein PDA $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$ mit akzeptierenden Zuständen und eine Konfiguration (q, ε, u) mit $q \in Q$ und $u \in \Gamma^+$
Frage: Gibt es ein $w \in \Sigma^*$, sodass $(s, w, \tau_0) \vdash^* (q, \varepsilon, u)$ gilt?

Problem: PDA-Nonemptyness
Gegeben: Ein PDA $\mathcal{A}$ mit akzeptierenden Zuständen
Frage: Gilt $L(\mathcal{A}) \neq \varnothing$?

b) Aus der Vorlesung ist bekannt, dass PDA-Nonemptyness entscheidbar ist. Begründen Sie, dass auch PDA-Config-Reach entscheidbar ist, indem Sie PDA-Config-Reach auf PDA-Nonemptyness reduzieren. Begründen Sie kurz, dass es sich um eine Reduktion handelt und warum dann folgt, dass PDA-Config-Reach entscheidbar ist. Es ist weder eine formale Konstruktion noch ein Beweis, wie er in Aufgabenteil a) gefordert ist, notwendig. **(2 Punkte)**

[1] Das heißt, der String wird mit $\sqcup$ überschrieben.

Zusatzaufgabe [First come, first serve] **5 Punkte**

Kellerautomaten sind endliche Automaten mit einem zusätzlichen Keller, in dem Informationen gespeichert werden können und der nach dem LIFO-Prinzip arbeitet (Last In, First Out). Hier soll nun, in geeigneter Weise, analog zur Definition von Kellerautomaten, ein *Warteschlangen-Automat* definiert werden, der statt eines Kellers mit einer Warteschlange (*Queue*) ausgestattet ist. Eine solche Warteschlange arbeitet nach dem FIFO-Prinzip (First In, First Out), d.h. Zeichen, die zuerst in die Warteschlange geschrieben wurden, sollen auch zuerst wieder gelesen werden.

a) Definieren Sie die Syntax und die Semantik eines solchen Automaten. Er soll in jedem Schritt höchstens ein Zeichen der Eingabe lesen, genau ein Zeichen aus der Warteschlange holen und beliebig viele Zeichen an die Warteschlange anfügen können. Außerdem soll er über akzeptierende Zustände verfügen. **(2 Punkte)**

b) Zeigen Sie: Zu jeder Sprache L, die von einer Turingmaschine entschieden wird, gibt es auch einen Warteschlangen-Automaten, der L entscheidet. **(3 Punkte)**

Testfragen

1. Gibt es eine Turingmaschine M, die nicht $L(M)$ entscheidet, obwohl M auf allen Eingaben hält?
2. Welche Aussage trifft die Church-Turing-These? Kann sie widerlegt oder bewiesen werden?
3. Kann mit Hilfe der Diagonalisierungstechnik direkt gezeigt werden, dass TM-Halt unentscheidbar ist?