Technische Universität München
Fakultät für Informatik
Prof. Tobias Nipkow, Ph.D.
Sascha Böhme, Lars Noschinski

Sommersemester 2011
Lösungsblatt 8
18. Juli 2011

Einführung in die Theoretische Informatik

Hinweis: *Bitte beachten Sie unbedingt die Hinweise zum Übungsablauf und zu den Aufgabentypen auf der THEO-Website (`http://theo.in.tum.de/`).*

Hausaufgabe 1 (5 Punkte)

Sei $\Sigma = \{0, 1\}$ und $L \subseteq \Sigma^*$. Zeigen Sie: Ist $M = (Q, \Sigma, \Gamma \cup \{\square\}, \delta, q_0, \square, F)$ eine deterministische Turingmaschine, die L akzeptiert, so gibt es auch eine Turingmaschine M' mit Bandalphabet $\Gamma' = \{0, 1, \square\}$, die L akzeptiert.

Lösungsvorschlag

Ist $|\Gamma| = 2$, so ist nichts zu tun. Ansonsten lässt sich jedes Zeichen aus Γ durch $c = \lceil \log_2 |\Gamma| \rceil$ Zeichen aus $\{0, 1\}$ darstellen ($c > 1$), indem die Zeichen aus Γ in beliebiger Reihenfolge durchnummeriert werden. Die c-stellige Binärdarstellung dieser Nummer wird dann verwendet, um dieses Zeichen zu kodieren. Wir gehen im Folgenden davon aus, dass 0 durch $0^{c-1}0$ und 1 durch $0^{c-1}1$ kodiert wird.
Zunächst müssen wir die Eingabe in dieses Format umwandeln. Dazu merken wir uns das erste Zeichen auf dem Band und löschen es. Dann bewegen wir den Kopf an das Ende der Eingabe, überspringen (als Trennzeichen) ein $\square$ und schreiben dort $c - 1$ mal eine 0, gefolgt von dem gemerkten Zeichen. Dies wiederholen wir jetzt (bewegen uns jetzt aber vor dem Schreiben ans Ende des Bandes, nicht nur hinter das erste Blank), bis die ursprüngliche Eingabe vollständig gelesen wurde. Den Kopf bewegen wir danach auf das erste Bandzeichen.
Nun müssen wir die Turingmaschine M simulieren. Dazu verarbeiten wir die Eingabe in Blöcken der Größe c und merken uns weiterhin im Zustand unserer Simulationsmaschine den Zustand der simulierten Maschine M (anfänglich q_0). Dies ist möglich, da Q endlich ist. Ein Schritt von M wird wie folgt simuliert:
Ist $q \in F$ so wechselt die Simulation in einen Endzustand und terminiert. Andernfalls liest die Simulation c Zeichen vom Band ein, merkt sich diese Zeichen im Zustand (dies ist möglich, da c und Γ' endlich sind) und bewegt den Kopf wieder zurück. Nun kennen wir den Zustand q von M und das durch die c Zeichen kodierte Zeichen x. Ist $\delta(q, x)$ nicht definiert, so bricht die simulierende Maschine nicht-akzeptierend ab.
Sei nun $\delta(q, x) = \{(q', x, Y)\}$. Jetzt wird die Kodierung des Zeichens x' aufs Band geschrieben und der Kopf zurückbewegt. Je nachdem, ob Y jetzt L, N oder R ist, wird der Kopf c Zeichen nach links, gar nicht oder c Zeichen nach rechts bewegt und wir merken uns q' als Zustand der simulierten Maschine. Danach simulieren wir den nächsten Schritt.

Hausaufgabe 2 (5 Punkte)

Beweisen Sie: Eine Turingmaschine, die ihren Lese-/Schreibkopf nur nach rechts bewegt, akzeptiert eine reguläre Sprache.

Hinweis: Überführen Sie die Turingmaschine zunächst in eine Form, so dass sie die Eingabe vollständig liest und darüberhinaus nicht auf das Band zugreift. Beachten sie auch nichtterminierende Läufe der Turingmaschine.

Lösungsvorschlag

Wir machen zunächst eine Reihe von Vorüberlegungen zu einer Turingmaschine A, die ihren Lese-/Schreibkopf nur nach rechts bewegt.

- Wegen Satz 4.11 können wir annehmen, dass A deterministisch ist.
- Da eine Turingmaschine sofort akzeptiert, wenn ein Endzustand erreicht wurde, können wir annehmen, dass F aus genau einem Zustand q_f besteht (ist F leer, so erweitern wir F und Q einfach um einen unerreichbaren Zustand).
- Eine Konfiguration von A kann vereinfacht durch (q, w) beschrieben werden, denn alle Zeichen auf dem Band links vom Lese-/Schreibkopf können keine Transitionen mehr beeinflussen.
- Wenn der Lese-/Schreibkopf von A auf dem ersten Leerzeichen $\square$ steht, dann terminiert A entweder sofort oder geht sofort in einen Endzustand über. Andernfalls können wir A wie folgt transformieren. Dabei ist die Konfiguration ab dem ersten Leerzeichen immer von der Form (q, ϵ), und wir müssen nur Übergänge der Form $\delta(q, \square) = \ldots$ betrachten.
  - Gibt es eine Folge von Zuständen $q_1, \ldots, q_n$ mit $q_1 = q_n$ und kein q_i ist ein Endzustand sowie $\delta(q_i, \square) = (q_{i+1}, a_{i+1}, R)$ für alle $1 \leq i < n$, so machen wir $\delta(q_i, \square)$ undefiniert für alle obigen i (denn ein Endzustand kann nicht erreicht werden).
  - Ist $\delta(q_1, \square) = (q_2, a, R)$ und $\delta(q_2, \square) = (q_3, b, R)$ und q_2 kein Endzustand, so setzen wir $\delta(q_1, \square) = (q_3, b, R)$.

  Nach diesen Transformationen terminiert A spätestens nach dem Lesen des ersten Leerzeichens und akzeptiert ein Wort genau dann, wenn die ursprüngliche TM dies macht.
- A durchläuft das gesamte Eingabewort w, falls $w \in L(A)$, das heißt, jeder Endzustand kann nur durch das Lesen eines Leerzeichens erreicht werden. Ist dies nicht so, fügen wir zusätzliche Transitionen ein, so dass in allen anderen Fällen A das Band bis zum Ende durchläuft (ohne die Eingabe zu beachten), und beim ersten Blank akzeptiert.

Wir können eine solche Turingmaschine $A = (Q, \Sigma, \Gamma, \delta, q_0, \square, q_f)$, die ihren Lese-/Schreibkopf nur nach rechts bewegt, durch einen DFA $A' = (Q \cup \{q_s\}, \Sigma, \delta', q_0, F')$ simulieren, wobei wir

$$\delta'(q, a) = \begin{cases} q' & \text{falls } \delta(q, a) = (q', b, R) \text{ für ein } b \in \Sigma \\ q_s & \text{sonst} \end{cases}$$
$$\delta'(q_s, a) = q_s$$

für alle $a \in \Sigma$ und $F' = \{q \in Q \mid \delta(q, \square) \in F\}$ setzen.

Man kann dann durch Induktion über die Länge von $w \in \Sigma^*$ zeigen, dass für all $q, q' \in Q$ die Gleichung $\hat{\delta}'(q, w) = q'$ genau dann gilt, wenn $(q, w) \to_A^* (q', \epsilon)$.
Dann gilt:

$$\begin{aligned} L(A') &= \{w \in \Sigma^* \mid \hat{\delta}'(q_0, w) \in F\} \\ &= \{w \in \Sigma^* \mid \exists q' \in F.\ (q_0, w) \to_a^* (q', \epsilon) \to_A \{q_f\}\} \\ &= L(A) \end{aligned}$$

Da A' die gleiche Sprache wie A akzeptiert, ist $L(A)$ regulär.

Hausaufgabe 3 (5 Punkte)

1. Zeigen Sie durch Rückführung auf die Definition, dass folgende Funktionen primitiv-rekursiv sind:

   (a) $\mathit{iszero}\,(x) = \begin{cases} 1 & \text{falls } x = 0 \\ 0 & \text{sonst} \end{cases}$

   (b) $\mathit{eq}\,(x,\, y) = \begin{cases} 1 & \text{falls } x = y \\ 0 & \text{sonst} \end{cases}$

   Sie dürfen bei den Definitionen der Funktionen nur auf die in Definition 4.30 genannten Funktionen sowie die Addition $x + y$ und $\mathit{pred}\,(x)$ und alle auf diesem Übungsblatt als PR bewiesenen Funktionen zurückgreifen. Außerdem dürfen Sie das erweiterte Schema der primitiven Rekursion verwenden.

2. Zeigen Sie durch Rückführung auf die Definition, dass die Division natürlicher Zahlen primitiv-rekursiv ist. Sie dürfen dazu das erweiterte Schema der primitiven Rekursion und die erweiterte Komposition verwenden sowie alle Funktionen (insbesondere arithmetische), die in der Vorlesung sowie auf diesem Übungsblatt als PR gezeigt wurden. Wenn Sie weitere Funktionen verwenden, dann müssen Sie für sie zunächst zeigen, dass sie PR sind.

Lösungsvorschlag

1. (a)
$$\begin{aligned} \mathit{iszero}\,(0) &= 1 \\ \mathit{iszero}\,(x+1) &= 0 \end{aligned}$$

   (b)
$$\begin{aligned} \mathit{eq}\,(x,\, y) &= \mathit{iszero}\,(\mathit{add}(\mathit{sub}(x, y), \mathit{sub}(y, x))) \\ \mathit{sub}\,(0,\, y) &= y \\ \mathit{sub}\,(x+1,\, y) &= \mathit{pred}\,(\mathit{sub}\,(x, y)) \end{aligned}$$

2.
$$\begin{aligned} \mathit{div}\,(x,\, y) &= \mathit{max}\,\{q \leq x \mid \mathit{lesseq}\,(q * y, x)\} \\ \mathit{lesseq}\,(x,\, y) &= \mathit{ifthen}\,(x \dot{-} y,\, \mathbf{false},\, \mathbf{true}) \end{aligned}$$

Hausaufgabe 4 (5 Punkte)

Wir betrachten zwei sich gegenseitig aufrufende, rekursive Funktionen. Diese seien wie folgt definiert

$$f(0) = s_0 \qquad f(m+1) = s \qquad g(0) = t_0 \qquad g(m+1) = t$$

wobei

- s_0 und t_0 aus primitiv-rekursiven Funktionen und
- s und t aus primitiv-rekursiven Funktionen sowie $f(m)$ und $g(m)$

zusammengesetzt sind. Zeigen Sie, dass f und g primitiv rekursiv sind. Sie dürfen dazu die erweiterte PR-Syntax und die Funktionen c, p_1 und p_2 aus der Vorlesung verwenden. *Hinweis:* Konstruieren Sie eine Funktion h, die $f(n) = p_1(h(n))$ und $g(n) = p_2(h(n))$ erfüllt.

Lösungsvorschlag

Wir konstruieren eine Funktion h wie folgt:

$$h(0) = c(s_0, t_0) \qquad h(m+1) = c(s', t')$$

wobei s' bzw. t' aus s bzw. t entstehen, indem $f(m)$ und $g(m)$ durch $p_1(h(m))$ und $p_2(h(m))$ ersetzt werden. Dann ist h primitiv rekursiv, denn es entspricht der erweiterten PR-Syntax. Die Funktionen f und g ergeben sich jetzt als Projektionen von h, nämlich $f(m) = p_1(h(m))$ und $g(m) = p_2(h(m))$ und sind daher ebenfalls primitv rekursiv.

Quiz 1

Ist die folgende Funktion primitiv-rekursiv?

$$f(n) = \begin{cases} n - 10 & \text{falls } n > 100 \\ f(f(n + 11)) & \text{sonst} \end{cases}$$

Lösungsvorschlag

Ja, denn f hat die folgende primitiv-rekursive Definition:

$$f(n) = \mathit{ifthen}(n > 100,\ n - 10,\ 91)$$

wobei wir annehmen, dass $\mathit{ifthen}(n, a, b)$ und $x > y$ primitiv rekursiv sind (siehe Tutoraufgabe 1 und Hausaufgabe 3).

Tutoraufgabe 1

Zeigen Sie durch Rückführung auf die Definition, dass die folgenden Funktionen primitiv rekursiv sind.

1. $\mathit{twopow}(n) = 2^n$
2. $\mathit{tower}(n) = 2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}$ (d.h. $2^{(2^{(2^{\cdot^{\cdot^{\cdot^2}}})})}$, Turm der Höhe n)
3. $\mathit{ifthen}(n, a, b)$ mit
$$\mathit{ifthen}(n, a, b) = \begin{cases} a & n \neq 0 \\ b & n = 0 \end{cases}$$

Sie dürfen dabei das erweiterte Schema der primitiven Rekursion sowie alle bisher definierten Funktionen (insbesondere numerische Konstanten sowie die Funktionen $\mathit{add}(x, y)$ und $\mathit{mult}(x, y)$) verwenden.

Lösungsvorschlag

Die Konstante 1 und die Multiplikation $\mathit{mult}(x, y)$ ist nach Vorlesung PR. Wir benutzen außerdem das erweiterte Schema der primitiven Rekursion.

1. $$\begin{aligned} \mathit{twopow}(0) &= 1 \\ \mathit{twopow}(n + 1) &= \mathit{mult}(\mathit{twopow}(n), 2). \end{aligned}$$
2. $$\begin{aligned} \mathit{tower}(0) &= 1 \\ \mathit{tower}(n + 1) &= \mathit{twopow}(\mathit{tower}(n)). \end{aligned}$$
3. $$\begin{aligned} \mathit{ifthen}(0, a, b) &= b \\ \mathit{ifthen}(n + 1, a, b) &= a. \end{aligned}$$

Tutoraufgabe 2

Sei $f : \mathbb{N} \to \mathbb{N}$ diejenige Funktion, die für alle $n \in \mathbb{N}, n \neq 0$ durch die Rekursion

$$f(n+1) = f(n) \cdot f(n-1)$$

mit den Startwerten $f(0) = 1$ und $f(1) = 2$ definiert ist.

1. Zeigen Sie, dass f primitiv-rekursiv ist, indem Sie ein `LOOP`-Programm angeben.
2. Zeigen Sie durch Rückführung auf die Definition, dass f primitiv-rekursiv ist.

Lösungsvorschlag

1. Das folgende `LOOP`-Programm basiert auf den Variablen $x_0, \ldots, x_5$. In x_0 wird das Ergebnis $f(n)$ ausgegeben, x_1 enthält beim Start das Argument n.

   $x_1 := n;$
   $x_2 := x_1 - 1;$
   $x_3 := 1;$
   $x_4 := 2;$
   $LOOP\ x_2\ DO$
   $\quad x_5 := x_4 * x_3;$
   $\quad x_3 := x_4;$
   $\quad x_4 := x_5$
   $END;$
   $x_0 := x_5$
   $IF\ x_1 = 0\ THEN\ x_0 := 1\ END;$
   $IF\ x_1 = 1\ THEN\ x_0 := 2\ END$

2. Wir wollen nun einen alternativen Beweis der PR von f geben, der den Beweis des Satzes 4.41 erhellt. Wir stützen uns dabei auf die Definition der primitiven Rekursivität und des erweiterten Schemas zur Definition primitiv rekursiver Funktionen.

   Die Schwierigkeit der Anwendung des Schemas der primitiven Rekursion auf die Funktion f besteht darin, dass die Funktionswerte $f(n+1)$ nicht allein von $f(n)$ sondern auch von $f(n-1)$ abhängen. Dieses Problem kann man angehen, in dem man Tupel von natürlichen Zahlen als natürliche Zahl kodiert, etwa so wie das in dem Beweis von Satz 4.41 gemacht wird. Wir simulieren aber nicht das `LOOP`-Programm, wie in dem Beweis von 4.41, sondern wir extrahieren die wichtigsten Schritte.

   Um nur auf einen vorherigen Funktionswert zurückgreifen zu müssen, definieren wir eine Funktion $g(n) = c(f(n), f(n+1))$ und zeigen, dass diese primitiv rekursiv ist. Wegen $f(n) = p_1(g(n))$ ist dann auch $f(n)$ primitiv rekursiv. Die Rekursion für g liest sich wie folgt. Wir haben $g(0) = c(1, 2)$ und (wenn wir $c(x, y)$ als $\binom{x}{y}^T$ schreiben):

   $$g(n+1) = \begin{pmatrix} f(n+1) \\ f(n+2) \end{pmatrix}^T = \begin{pmatrix} f(n+1) \\ f(n+1) \cdot f(n) \end{pmatrix}^T = \begin{pmatrix} p_2(g(n)) \\ p_2(g(n)) \cdot p_1(g(n)) \end{pmatrix}^T$$

   Dies folgt dem erweiterten Rekursionsschema für primitive Rekursivität. Also ist g und damit auch f primitiv rekursiv.

Tutoraufgabe 3

Für eine beliebige Funktion $f : \mathbb{N} \to \mathbb{N}$ betrachten wir die Funktion $U_f : \mathbb{N} \to \mathbb{N}$ mit

$$U_f(n) = \begin{cases} \min\{m \in \mathbb{N} \mid f(m) = n+1\} & \text{falls ein solches } m \text{ existiert} \\ \bot & \text{sonst} \end{cases}$$

Zeigen Sie: Wenn f total und μ-rekursiv ist, dann ist auch U_f μ-rekursiv.

Lösungsvorschlag

Wir verwenden den μ-Operator, um die Funktion U_f zu definieren, die die „Umkehrfunktion“ von f ist. Wir definieren eine Hilfsfunktion $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, mit

$$h(m,n) = f(m) \dot{-} (n+1) + (n+1) \dot{-} f(m)$$

Die Definition ist gerade so gewählt, dass $h(m,n) = 0 \iff f(m) = n+1$ gilt, denn $h(m,n) = |f(m) - (n+1)|$. Außerdem ist h total, da f total ist.
Dann ist $U_f = \mu h$, und es gilt

$$\begin{aligned} U_f(n) &= \begin{cases} \min\{m \in \mathbb{N} \mid h(m,n) = 0\} & \text{falls ein solches } m \text{ existiert} \\ & \text{und } \underbrace{h(k,n) \neq \bot \text{ für alle } k \leq m}_{\text{immer erfüllt, da } h \text{ total ist}} \\ \bot & \text{sonst} \end{cases} \\ &= \begin{cases} \min\{m \in \mathbb{N} \mid f(m) = n+1\} & \text{falls ein solches } m \text{ existiert} \\ \bot & \text{sonst.} \end{cases} \end{aligned}$$