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

Sommersemester 2011
Übungsblatt 7
27. Juni 2011

Einführung in die Theoretische Informatik

Abgabetermin: 4. Juli 2011 **vor der Vorlesung** *in die THEO-Briefkästen*

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

Hausaufgabe 1 (8 Punkte)

Sei $\Sigma = \{0, 1\}$. Wie in Übung 5 bezeichnen wir mit $\overline{w}$ die Negation eines Wortes $w \in \{0, 1\}^*$, d.h. alle Nullen werden durch Einsen ersetzt und umgekehrt.

1. Geben Sie eine deterministische Turingmaschine an, die für ein Eingabewort $w \in \Sigma^*$ folgende Berechnung durchführt: Am Ende der Berechnung steht auf dem Band das Wort $w\overline{w}$ und der Kopf steht in einem Endzustand auf dem ersten Zeichen dieses Wortes.

   Kommentieren Sie ihre Konstruktion durch eine informelle Beschreibung Ihrer Lösungsidee.

2. Geben Sie nun eine Turingmaschine an, die die Sprache $L = \{w\overline{w} \mid w \in \Sigma^*\}$ akzeptiert. Kommentieren Sie wiederum ihre Konstruktion durch eine informelle Beschreibung Ihrer Lösungsidee.

Hausaufgabe 2 (6 Punkte)

1. Zeigen Sie, dass das if-then-else-Konstrukt in `LOOP`-Programmen wirklich nur eine syntaktische Abkürzung ist. Geben Sie dazu zu `IF` $X = 0$ `THEN` P `ELSE` Q `END` ein `LOOP`-Programm an, dass nur aus den vier primitiven Operationen besteht.

2. Zeigen Sie jetzt, dass statt einer atomaren Bedingung $X = 0$, wobei X eine Variable ist, auch beliebige Verknüpfungen solcher atomaren Bedingungen mit $\neg$ (Negation), $\wedge$ (Konjunktion) und $\vee$ (Disjunktion) verwendet werden können.

   Geben Sie dazu eine rekursive Funktion f an, die `IF` B `THEN` P `ELSE` Q `END` mit einer solchen Bedingung B als Eingabe nimmt und es in ein `LOOP`-Programm mit der gleichen Semantik umformt. In der Ausgabe dürfen $\neg$, $\wedge$ und $\vee$ nicht mehr vorkommen (wohl aber if-then-else).

   Beispiel: Die Transformationsregel für Negation lässt sich wie folgt definieren.

$$f(\texttt{IF } \neg B \texttt{ THEN } P \texttt{ ELSE } Q \texttt{ END}) = f(\texttt{IF } B \texttt{ THEN } Q \texttt{ ELSE } P \texttt{ END})$$

3. Wir erweitern die Bedingungen aus dem ersten Teil, so dass zusätzlich Tests der Form $E = 0$ erlaubt sind, wobei E ein beliebiger arithmetischer Ausdruck aus Variablen, natürlichen Zahlen, $+$ und $*$ ist.

   Erweitern Sie die Funktion f, so dass sie auch solche Bedingungen auf elementare `LOOP`-Programme abbildet, d.h., arithmetische Ausdrücke kommen in der Ausgabe nicht mehr vor.

Hausaufgabe 3 (6 Punkte)

Wir bezeichnen mit TM_k solche Einband-Turingmaschinen, die jede Zelle des Bandes höchstens k-mal ändern dürfen. Dabei gelten nur Übergänge $\delta(q, x) = (q', y, X)$ mit $x \neq y$ als Änderungen einer Zelle des Bandes (mit $X \in \{N, R, L\}$).

1. Zeigen Sie, dass die Turingmaschinen TM_2 äquivalent zu herkömmlichen Turingmaschinen sind. Benutzen Sie soviel Band wie nötig.

2. Zeigen Sie, dass auch die Turingmaschinen TM_1 äquivalent zu herkömmlichen Turingmaschinen sind. Sie dürfen dabei die Resultate der ersten Teilaufgabe verwenden.

Sie müssen keine expliziten Konstruktionen angeben. Es genügen informelle, aber dennoch vollständige und genaue Beschreibungen.

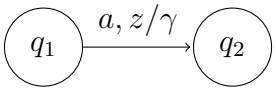
Quiz 1

Beantworten Sie kurz die folgenden Fragen:

1. Gibt es eine Turingmaschine, die sich nie mehr als vier Schritte vom Startzustand entfernt und eine unendliche Sprache akzeptiert?
2. Welche Sprachen lassen sich mit Turingmaschinen, die ihren Kopf immer nur nach rechts bewegen, erkennen?
3. Vergleichen Sie die beiden folgenden `WHILE`-Programme: `WHILE` $b_0 \wedge b_1$ `DO` P `END` und `WHILE` b_0 `DO` P `END;` `WHILE` $b_0 \wedge b_1$ `DO` P `END`. Zeigen diese Programme das gleiche Verhalten?

Tutoraufgabe 1

Ein *Queue-Automat* (kurz: QA) ist ein Tupel $A = (Q, \Sigma, \Gamma, q_0, z_0, \delta)$. Dabei sind Q (Zustandsmenge), Σ (Eingabealphabet) und Γ (Queuealphabet) endliche Mengen ähnlich wie bei einer Turingmaschine, $q_0 \in Q$ ist der Startzustand, und $z_0 \in \Gamma$ beschreibt die initiale Queue. Die Übergangsfunktion δ hat den Typ $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$. Graphisch kann eine Transition $(q_2, \gamma) \in \delta(q_1, a, z)$ eines Queue-Automaten wie folgt dargestellt werden:



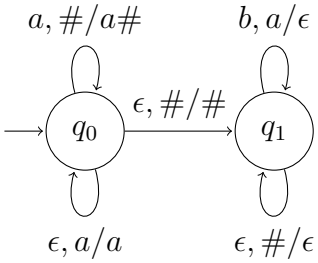
Eine *Konfiguration* eines Queue-Automaten ist ein Tripel $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$. Dabei ist q der aktuelle Zustand, w das noch zu lesende Wort und γ der aktuelle Inhalt der Queue. Daraus ergibt sich die Übergangsrelation $\to_A$ zwischen Konfigurationen:

$$(q, bw, z\gamma) \to_A (q', w, \gamma\gamma')\,, \qquad \text{wenn} \qquad (q', \gamma') \in \delta(q, b, z)\ .$$

Die (mit leerer Queue) akzeptierte Sprache eines Queue-Automaten ist

$$L_\epsilon(A) = \{w \in \Sigma^* \mid \exists q' \in Q.\ (q_0, w, z_0) \to_A^* (q', \epsilon, \epsilon)\}\ .$$

1. Der Queue-Automat $A_1 = (\{q_0, q_1\}, \{a, b\}, \{a, b, \#\}, q_0, \#, \delta)$ sei wie folgt graphisch dargestellt:



   Geben Sie eine *akzeptierende* Konfigurationsfolge für das Eingabewort *aabb* an.

2. Geben Sie einen Queue-Automaten A_2 an mit $L(A_2) = \{ww^R \mid w \in \{a, b\}^*\}$.
3. Zeigen Sie, dass jede (deterministische) Turingmaschine durch einen Queue-Automaten simuliert werden kann.

Tutoraufgabe 2

Wir wollen untersuchen, ob sich (ähnlich wie bei den WHILE-Programmen) auch jedes LOOP-Programm in eine „Normalform“

$$\texttt{LOOP } X \texttt{ DO } P \texttt{ END}$$

bringen lässt, so dass P keine Schleifen mehr enthält.
Wir bezeichnen mit $V(P)$ die Menge der Variablen, die in P vorkommen (diese Menge ist stets endlich). Für einen gegebenen Programmlauf bezeichnen wir mit $[x]$ den Wert der Variablen x beim Start des Programms, und mit $[x]'$ den Wert der Variablen nach Programmende.

1. Zeigen Sie: Für jedes LOOP-Programm P *ohne Schleifen* gibt es eine Konstante k, so dass gilt:
$$\max_{x \in V(P)} [x]' \leq \max_{x \in V(P)} [x] + k \; .$$

2. Zeigen Sie, dass es kein LOOP-Programm in Normalform geben kann, welches die Quadratfunktion $n \mapsto n^2$ berechnet.