Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
Dr. M. Lange

SS 05

30.5.05

Übungen zur Vorlesung
Informatik IV

Blatt 7
Abgabe spätestens am 6.6.05, 14:00 Uhr

Aufgabe 38: **4 Punkte**

Betrachten Sie den PDA $\mathcal{A} = (Z, \Sigma, \Gamma, \delta, z_0, \#)$, wobei $Z = \{z_0, z_1\}$, $\Sigma = \{a, b\}$, $\Gamma = \{\#, X\}$ und δ folgendermaßen gegeben ist.

$$\begin{aligned}
\delta(z_0, a, \#) &= \{(z_0, X\#)\} \\
\delta(z_0, a, X) &= \{(z_0, XX)\} \\
\delta(z_0, b, X) &= \{(z_0, X)\} \\
\delta(z_0, \epsilon, X) &= \{(z_1, \epsilon)\} \\
\delta(z_1, \epsilon, X) &= \{(z_1, \epsilon)\} \\
\delta(z_1, b, X) &= \{(z_1, XX)\} \\
\delta(z_1, b, \#) &= \{(z_1, \epsilon)\}
\end{aligned}$$

Wandeln Sie den PDA $\mathcal{A}$ mit dem in der Vorlesung vorgestellten Verfahren in eine äquivalente Typ-2-Grammatik um.

Aufgabe 39: **4 Punkte**

In der Vorlesung wurden Kellerautomaten vorgestellt, bei denen das Akzeptanz-Kriterium “leerer Keller” ist. D.h. ein Wort wird akzeptiert, genau dann wenn es einen Lauf des Automaten gibt, der nach vollständigem Lesen des Eingabeworts in einer Konfiguration mit leerem Keller endet. Analog zu NEAs kann man auch Kellerautomaten betrachten, bei denen Akzeptanz über das Erreichen eines Endzustands definiert ist. Dabei muss der Keller nicht geleert werden.

Ein *Endzustands-Kellerautomat* (EPDA) ist ein Tupel $\mathcal{A} = (Z, \Sigma, \Gamma, \delta, z_0, \#, E)$, so dass $E \subseteq Z$ und $Z, \Sigma, \Gamma, \delta, z_0$ und $\#$ wie bei einem herkömmlichen Kellerautomaten definiert sind. Die von einem EPDA $\mathcal{A}$ erkannte Sprache ist

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid (z_0, w, \#) \vdash^* (z, \epsilon, \gamma) \text{ für ein } z \in E \text{ und ein } \gamma \in \Gamma^*\}$$

Zeigen Sie, dass beide Automatenmodelle dieselbe Klasse von Sprachen erkennen. Dazu müssen Sie aus einem PDA einen äquivalenten EPDA und umgekehrt konstruieren.

Sie brauchen die Korrektheit Ihrer Konstruktionen nicht formal zu beweisen. Ihre Ideen müssen Sie aber erläutern!

Aufgabe 40: **6 Punkte**

Ein PDA $\mathcal{A} = (Z, \Sigma, \Gamma, z_0, \delta, \#, E)$ heisst *deterministisch* (DPDA), falls in jeder Konfiguration höchstens eine Möglichkeit besteht, zu einer Folgekonfiguration überzugehen. D.h. für alle $z \in Z$, alle $a \in \Sigma$, alle $A \in \Gamma$ gilt:

$$|\delta(z, a, A)| + |\delta(z, \epsilon, A)| \leq 1 \quad (1)$$

Darüberhinaus akzeptiert ein DPDA ein Wort durch Erreichen eines Endzustandes nachdem das Eingabewort vollständig gelesen wurde. (Ein DPDA ist also in der Terminologie der vorherigen Aufgabe ein EPDA.)

a) Geben Sie zu einem beliebigen DPDA $\mathcal{A}$ einen DPDA $\overline{\mathcal{A}}$ an, der das Komplement der von $\mathcal{A}$ erkannten Sprache erkennt. Zeigen Sie also: (1) Ist $w \in L(\mathcal{A})$, dann ist $w \notin L(\overline{\mathcal{A}})$, und (2) ist $w \in L(\overline{\mathcal{A}})$, dann ist $w \notin L(\mathcal{A})$.

   Hinweis: Es reicht nicht, einfach nur Endzustände und Nicht-Endzustände zu vertauschen, da ein DPDA auch eine Eingabe verwerfen kann, ohne Sie vollständig zu lesen. Ansonsten müsste Gleichheit in (1) bestehen.

b) Zeigen Sie, dass die Sprache L aller Wörter über dem Alphabet $\Sigma = \{a, b\}$, in denen man von links nach rechts gelesen niemals mehr bs als as sieht, von einem DPDA erkannt wird. Formal ist $L = \{w \in \Sigma^* \mid \forall v, u \in \Sigma^* : w = vu \Rightarrow |v|_b \leq |v|_a\}$.

c) Läßt sich ein beliebiger PDA in einen äquivalenten DPDA umwandeln, z.B. per Potenzmengenkonstruktion, wie es bei NEAs und DEAs möglich ist? Begründen Sie Ihre Antwort!

Aufgabe 41: **2 Punkte**

Gesucht ist nach einer Lösung für das Äquivalenzproblem für deterministische Kellerautomaten. Können Sie einen Algorithmus angeben, der zu zwei gegebenen DPDAs $\mathcal{A}$ und $\mathcal{B}$ entscheidet, ob $L(\mathcal{A}) = L(\mathcal{B})$ gilt? Begründen Sie Ihre Antwort!

Aufgabe 42: **4 Punkte**

Konstruieren Sie eine Turing-Maschine $\mathcal{A}$, die die Funktion "plus 3" berechnet. $\mathcal{A}$ geht davon aus, dass zu Beginn auf dem Eingabeband eine natürliche Zahl n in binärer Kodierung mit dem niedrigwertigsten Bit rechts steht. Am Ende der Berechnung, d.h. wenn ein Endzustand erreicht ist, soll dort $n + 3$ stehen.

Hinweis: Betrachten Sie die Turing-Maschine, die die Funktion "plus 1" berechnet (vgl. z.B. Buch von Schöning).