Informatik III - Blatt 8

Aufgabe 1:

a) Es soll gelten $\exists m_i \neq m_j$ für ein $i \neq j$.

$\Rightarrow m_i \neq m_1$ oder $m_j \neq m_1$ (denn die sind eben ungleich)

$\Rightarrow$ den ersten Block von a's (a^{m_1}) einkellern:

$q_0 a \# \to q_0 A \#$, $q_0 a A \to q_0 AA$

$q_0 b \# \to q_1 \# \mid q_2 \#$, $q_0 bA \to q_1 A \mid q_2 A$

nächster Block a's ist gleich lang wie a^{m_1} $\Rightarrow$ überspringen

$\Rightarrow q_1$: den nächsten a-Block überspringen:

Sei $k = \{A, \#\}$.

$q_1 a k \to q_1 k$, $q_1 b k \to q_1 k \mid q_2 k$ ← jetzt kommt der richtige

nochmal übersp

$\Rightarrow q_2$: checken, ob für das nun kommende auch $a^{m_j} \neq a^{m_1}$.

$q_2 a A \to q_2 \varepsilon$ „immer einen weg von den a^{m_1}"

$q_2 a \# \to q_3 \#$ „$m_j > m_1 \Rightarrow$ gehe q_3, um zu akzeptieren"

$q_2 bA \to q_3 A$ „$m_j < m_1 \Rightarrow$ ~ "

$q_2 \varepsilon A \to q_3 A$ „$m_j < m_1$ und a^{m_j} = letzte a-Block"

$\Rightarrow$ Fertig und sagen q_3 ist Endzustand oder:

$q_3 a k \to q_3 k$, $q_3 b k \to q_3 k$

$q_3 \varepsilon A \to q_3 \varepsilon$

$q_3 \varepsilon \# \to q_3 \varepsilon$ $\Rightarrow$ der Keller wird leer gemacht und somit akzeptiert

b) Strategie: (Nichtdeterministisch)

Für jedes a ein B einkellern und dann für jedes b mindestens ein, vielleicht aber auch zwei B's auskellern.

$K = \{B, \#\}$

$\delta(z_0, a, K) = (z_0, BK)$ „Für jedes a ein B"

$\delta(z_0, a, K) = (z_1, BK)$ „beim letzten a auf z_1"

$\delta(z_1, b, B) = (z_1, \varepsilon) \mid (z_2, \varepsilon)$ „Für jedes b ein oder zwei B raus"

$\delta(z_2, \varepsilon, B) = (z_1, \varepsilon)$

$\delta(z_1, \varepsilon, \#) = (z_1, \varepsilon)$ „hier wird mit leerem Keller akzeptiert"

$\tilde{b}$) erkennt $\#a \leq \#b \leq 2\#a$

Strategie: sichere zuerst die a's auf dem Keller und stelle dann sicher, dass mindestens genau so viele, höchstens aber doppelt so viele b's folgen.

Sei $k = \{B, \#\}$

$\delta(z_0, a, k) = (z_0, Bk)$

$\delta(z_0, a, k) = (z_1, Bk)$ das letzte a erfordert Zustandsübergang

$\Rightarrow$ der Trick:

$\delta(z_1, \varepsilon, B) = (z_1, b) \mid (z_1, bb)$ damit für jedes a mind. 1 höchstens 2 b's.

$\delta(z_1, b, b) = (z_1, \varepsilon)$

c) $\#b \leq \#a \leq 2\#b$. Sei $k = \{\#, A\}$

$q_0\, aK \rightarrow q_0\, AK$ a's auf den Keller

$q_0\, bA \rightarrow q_0 \varepsilon \mid q_1 \varepsilon$ 1 (q_0) oder 2 (q_1) a's aus dem Keller weg

$q_1 \varepsilon A \rightarrow q_0 \varepsilon$ noch ein a raus

$q_1 \varepsilon \# \rightarrow q_0 B\#$ kein a mehr da $\Rightarrow$ b rein

$q_0\, b\# \rightarrow q_0 B\# \mid q_0 BB\#$ ein oder zwei b's rein, die jeweils

$q_0\, bB \rightarrow q_0 BB \mid q_0 BBB$ wieder durch ein a aufgewogen werden

$q_0\, aB \rightarrow q_0 \varepsilon$ b's aufwiegen

$q_0 \varepsilon \# \rightarrow q_0 \varepsilon$ $\Rightarrow$ Fertig, Keller leer.

d) Was heisst binär ·3?

Shifte die Zahl einmal nach rechts und addiere das darauf: $1000_2 \cdot 3_{10} = 1000_2 \cdot 11_2$ $(8 \cdot 3 = 24)$

$$
\begin{array}{r}
1000 \\
1000 \\
\hline
11000
\end{array}
$$

Der Automat sichert erstmal alles vor dem f auf dem Keller:

$q_0 0 k \to q_0 0 k$ mit $k = \{\#, 0, 1\}$

$q_0 1 k \to q_0 1 k \Leftrightarrow \delta(q_0, 1, k) = (q_0, 1k)$

Kommt das f springe ich auf q_1:

$q_0 f k \to q_1 \cancel{k}$

Die Zustände haben jetzt folgende Bedeutung:

q_1: kein Übertrag von vorher

q_2: entweder eine gemerkte 1 oder Übertrag

q_3: sowohl gemerkte 1, als auch Übertrag

q_1: $\delta(q_1, 0, 0) \to (q_1, \varepsilon)$ Bsp. 1000 f 00011
Im Original stand da eine 0 und es gibt keinerlei Überträge => o.k.

$\delta(q_1, 1, 1) \to (q_1, \varepsilon)$ Bsp. 1000 f 00011

q_2: $\delta(q_2, 1, 0) \to (q_1, \varepsilon)$ habe Übertrag bzw. gemerkte 1 und die steht jetzt da, wo im Original eine 0 war => passt

$\delta(q_2, 0, 1) \to (q_3, \varepsilon)$ hier steht eine Null, ich hätte aber lieber eine 1 => gehe zu Übertrag + gemerkte 1

$\delta(q_2, 1, \#) \to (q_2 \varepsilon)$ Fertig, da diese 1 gerade die im Original fehlende war...

q_3: $\delta(q_3, 0, 0) \to (q_2 \varepsilon)$ es ist eine 1 zu merken, weil $\overset{1}{0} = 10\ldots$

$\delta(q_3, 1, 1) \to (q_3 \varepsilon)$ es konnte noch keine der zwei zu verarbeitenden Einsen abgebaut werden

$\delta(q_3, 0, \#) \to (q_2, \#)$ wenn jetzt noch 1 kommt => Ziel...

Aufgabe 2:

Der Zustand des Kellerautomaten wird gerade vom Eingabezeichen und dem aktuellen Kellerinhalt bestimmt.
Es gibt nur endlich viele Keller- und Eingabezeichen und der Keller beinhaltet immer $\leq k$ Zeichen $\Rightarrow$ Eingabez. $\times$ $\mathbb{P}$(Kellerz.^{k}) sind alle möglichen Zustände und mit $|\text{Eingabez}| \cdot 2^{|\text{Kellerz}| \cdot k} < \infty$ gibt es den Automaten ohne Keller, der äquivalent ist $\Rightarrow$ regulär.

Die Übergangsfunktion δ ergibt sich folgendermaßen:

$$\underbrace{\delta(q, a, A) \to (q', \alpha)}_{\text{Automat}} \quad \Rightarrow \underbrace{\delta((q, A\gamma), a) = (q', \alpha\gamma)}_{\text{reguläre Grammatik}}$$

Ein Zustand im Automat q mit Kellerinhalt $A\gamma$ ergibt in der Grammatik Zustand $(q, A\gamma)$.

Startzustand der Grammatik ist der Startzustand mit dem Kellerbodenzeichen des Automaten, also $(q_s, \#)$.

Endzustand ist jeder Zustand mit leeren Keller bei Automaten, die durch leeren Keller akzeptieren (diese Zustände müssen nicht alle durch die Übergangsfkt. erreicht werden) bzw. diejenigen Zustände in der Grammatik, die den Endzustand des Automaten enthalten.

Aufgabe 3:

a) Betrachten wir die Regeln:

$z_0 \, o \, \# \Rightarrow z_0 \#$ — wandert zu Beginn zum Stamm: ⇒ egal ⇒ nix kellern

$z_0 \, k \, \# \Rightarrow z_1 D \#$ — eine Katze ⇒ kellere ein D und gehe zu z_1 ≙ Katze gefunden, soviele D's wie auf dem Keller − 1 darf der Ast darüber höchstens lang sein

$z_1 \, o \, D \Rightarrow z_1 D D$ — ein weiterer Sicherheitsabstand D kommt hinzu

$z_1 \, k \, D \Rightarrow z_3 D$
$z_3 \, \varepsilon \, D \Rightarrow z_3 \varepsilon$
$z_3 \, \varepsilon \, \# \Rightarrow z_1 D \#$
— Du kannst alle gemerkten D's wegschmeissen, weil da eine neue Katze ist. ← 1 Katze da, erstes D...

$z_1 \, s \, D \Rightarrow z_2 D$ — Stamm erreicht, jetzt darf der darüber nur Ds−1 lang sein...

$z_2 \, u \, D \Rightarrow z_2 \varepsilon$ — für jedes u ein D weg

$z_2 \, e \, D \Rightarrow z_4 D$ — mindestens das D von der Katze muss am Ende noch da sein...

Hugo verwirft also dann wenn nicht genügend D's eingelagert wurden, um in z_4 zu kommen und das ist genau dann der Fall, wenn die Katze unter dem Ast ist.

b) Mache das ganze nichtdeterministisch, d.h. springe erst in z_1, wenn Du bei der letzten Katze vor dem Stamm bist: $z_0 o \# \Rightarrow z_0 \#$, $z_0 k \# \Rightarrow z_0 \# \mid z_1 D \#$
(nicht die letzte) (die letzte)

Alle Löschregeln mit z_3 fallen somit auch weg.
⇒ ε-frei

c) in z_4 muss dann einfach noch der Keller von den D's befreit werden:

$z_4 \varepsilon D \to z_4 \varepsilon$
$z_4 \varepsilon \# \to z_4 \varepsilon$

d) $S \Rightarrow oS \mid kS$ alles vor der letzten Kote ist egal
$S \Rightarrow kS'$ die letzte Kote
$S' \Rightarrow oS' \mid Ae$ ← kann weiter weg vom Stamm sein, als der Ast darüber lang ist,
$A \rightarrow oAu \mid s$ ← muss aber mindestens so lang sein

e) Er muss dann auf seinem weg nicht nur D's für „ein Stück Weg gefunden" sondern auch K's für „eine Kote gefunden" einkellern.
Sein Weg auf dem Ast darüber muss dann beendet sein, bevor er wieder ein k aus dem Keller holt.

$z_0 o \bar{k} \to z_0 D \bar{k}$ je $\forall \bar{k} \in \{\#, D, k\}$
$z_0 k \bar{k} \to z_1 k \bar{k}$ mindestens eine Kote muss gefunden werden!

$z_1 o \bar{k} \to z_1 D \bar{k}$
$z_1 k \bar{k} \to z_1 k \bar{k}$
$z_1 s \bar{k} \to z_2 \bar{k}$ hoch
$z_2 u D \to z_2 \varepsilon$
$z_2 e D \to z_3 D$
$z_2 e k \to z_3 k$ z_3 ist akzeptierender Zustand.

f) Es geht nicht, denn nehmen wir an, es ginge, dann passiert Folgendes:

Die Zustandsmenge ist endlich. Sie sei n.
Der Automat ist determinist., d.h. bei jeder Konfigurations-Folge kann ich zu jedem gelesenen Zeichen genau sagen, wo ich mich im Automat (Zustand) befinde und was auf dem Keller ist.
Da ich nur D's auf dem Keller speichern kann, gibt es ein Problem, wenn nach dem Speichern einiger D's plötzlich wieder eine Kerte kommt. Ohne ε-Übergänge kann ich die D's nicht einfach löschen, durch den Determinismus kann ich aber auch nicht sagen: „schlechter Versuch".

Also habe ich nur die Möglichkeit, in meinen Zuständen zu kodieren, wieviele D's relevant sind.
Betrachte ich nun ein Eingabewort mit mehr als n relevanten D's, also mit mehr D's, als ich Zustände habe:

$$ko^n ko^{n+x} su^n e \quad x \geq 1$$

Das Wort liegt in der Sprache, unser Automat kann es aber nicht erkennen, weil ihm schon vor Erreichen des s die Zustände ausgehen... ↯