Karlsruher Institut für Technologie

aifb
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren

16.02.2015

Lösung zur Klausur über den Stoff der Vorlesung „Grundlagen der Informatik II“ (90 Minuten)

Name: _____ Vorname: _____

Matr.-Nr.: _____ Semester: _____ (WS 2014/15)

Ich bestätige, dass ich die folgenden Angaben gelesen und mich von der Vollständigkeit dieses Klausurexemplars überzeugt habe (Seiten 1-25).

_____

Unterschrift des o. g. Klausurteilnehmers bzw. der o. g. Klausurteilnehmerin

Anmerkungen:

1. Legen Sie bitte Ihren Studierendenausweis bereit.
2. Bitte tragen Sie **Name**, **Vorname** und **Matr.-Nr.** deutlich lesbar ein.
3. Die folgenden **11 Aufgaben** sind vollständig zu bearbeiten.
4. Folgende Hilfsmittel sind zugelassen: **keine.**
5. Täuschungsversuche führen zum Ausschluss von der Klausur.
6. Unleserliche oder mit Bleistift geschriebene Lösungen können von der Klausur bzw. Wertung ausgeschlossen werden.
7. Die Bearbeitungszeit beträgt 90 Minuten.

Nur für den Prüfer :

1	2	3	4	5	6	7	8	9	10	11	-	-	-	-	-	gesamt
(9)	(8)	(10)	(8)	(8)	(9)	(10)	(7)	(6)	(10)	(5)						(90)

Aufgabenübersicht

Aufgabe 1 **9 Punkte**

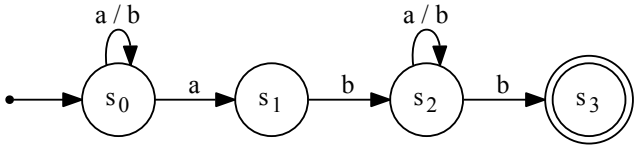
2015-H-01 **Endliche Automaten**

/ 9

Gegeben sei der folgende nichtdeterministische endliche Automat A mit

$$A = (\{a, b\}, \{s_0, \ldots, s_3\}, \delta, s_0, \{s_3\})$$

δ:



(a) Erstellen Sie einen **deterministischen** endlichen Automaten $A' = (E', S', \delta', s'_0, F')$ mit $L(A') = L(A)$ mithilfe des aus der Vorlesung bekannten Algorithmus. Nutzen Sie dafür die vorgegebene Tabelle. Definieren Sie A' **vollständig**. Geben Sie insbesondere ein Zustandsüberführungsdiagramm an.

/ 4

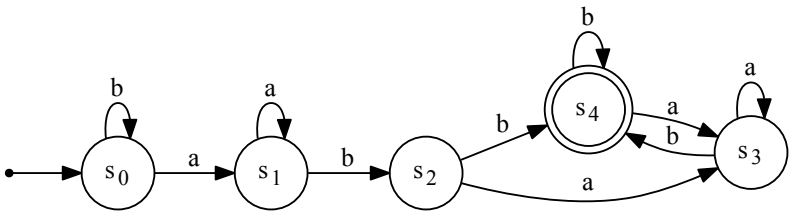
Lösung:

	a	b
$\{s_0\} \hat{=} s_0$	$\{s_0, s_1\} \hat{=} s_1$	$\{s_0\} \hat{=} s_0$
$\{s_0, s_1\} \hat{=} s_1$	$\{s_0, s_1\} \hat{=} s_1$	$\{s_0, s_2\} \hat{=} s_2$
$\{s_0, s_2\} \hat{=} s_2$	$\{s_0, s_1, s_2\} \hat{=} s_3$	$\{s_0, s_2, s_3\} \hat{=} s_4$
$\{s_0, s_1, s_2\} \hat{=} s_3$	$\{s_0, s_1, s_2\} \hat{=} s_3$	$\{s_0, s_2, s_3\} \hat{=} s_4$
$\{s_0, s_2, s_3\} \hat{=} s_4$	$\{s_0, s_1, s_2\} \hat{=} s_3$	$\{s_0, s_2, s_3\} \hat{=} s_4$

Hieraus ergibt sich der Automat

$$A' = (\{a, b\}, \{s_0, \ldots, s_4\}, \delta', s_0, \{s_4\})$$

δ':



mit der Zustandsüberführungstabelle:

	a	b
s_0	s_1	s_0
s_1	s_1	s_2
s_2	s_3	s_4
s_3	s_3	s_4
s_4	s_3	s_4

(b) Geben Sie einen zu A äquivalenten **deterministischen Kellerautomaten** A'' an, der lediglich **zwei Zustände** s_0'' und s_e verwendet. Geben Sie A'' vollständig an.

/ 5

Lösung:

$$A'' = (\{a, b\}, \{s_0'', s_e\}, \{k_0, a, b\}, \delta'', s_0'', k_0, \{s_e\})$$

$$\begin{aligned}
(s_0'', a, k_0) &\to (s_0'', a)\\
(s_0'', b, k_0) &\to (s_0'', k_0)\\
(s_0'', a, a) &\to (s_0'', a)\\
(s_0'', b, a) &\to (s_0'', b)\\
(s_0'', a, b) &\to (s_0'', b)\\
(s_0'', b, b) &\to (s_e, b)\\
(s_e, a, b) &\to (s_0'', b)\\
(s_e, b, b) &\to (s_e, b)
\end{aligned}$$

Alternativ können die Zustände des deterministischen endlichen Automaten A' über den Keller abgebildet werden, wobei k_0 den Startzustand s_0 symbolisiert (es wäre auch zulässig s_0 zusätzlich in das Kelleralphabet aufzunehmen und die entsprechenden Zustandsüberführungen zu definieren):

$$A'' = (\{a, b\}, \{s_0'', s_e\}, \{k_0, s_1, \ldots, s_4\}, \delta'', s_0'', k_0, \{s_e\})$$

$$\begin{aligned}
(s_0'', a, k_0) &\to (s_0'', s_1)\\
(s_0'', b, k_0) &\to (s_0'', k_0)\\
(s_0'', a, s_1) &\to (s_0'', s_1)\\
(s_0'', b, s_1) &\to (s_0'', s_2)\\
(s_0'', a, s_2) &\to (s_0'', s_3)\\
(s_0'', b, s_2) &\to (s_e, s_4)\\
(s_0'', a, s_3) &\to (s_0'', s_3)\\
(s_0'', b, s_3) &\to (s_e, s_4)\\
(s_e, a, s_4) &\to (s_0'', s_3)\\
(s_e, b, s_4) &\to (s_e, s_4)
\end{aligned}$$

Aufgabe 2 **8 Punkte**

2015-H-02 **Pumping-Lemma und Chomsky-Klassen**

/ 8

Gegeben sei die Sprache

$$L = \{a^m b^n c^{mn} \mid m, n \in \mathbb{N}_0\}$$

Es gilt beispielsweise:

- $\lambda, a, aa, aaa, \ldots; b, bb, bbb, \ldots; abc, aabcc, abbcc, aabbcccc \in L$,
- $ab, ac, bc, abcc, aabccc \notin L$.

(a) Zeigen Sie mit dem Pumping-Lemma für reguläre Sprachen, dass L nicht regulär ist.

/ 6

Lösung:

Angenommen, L wäre vom Typ 3. Dann existiert ein $n \in \mathbb{N}$, sodass jedes Wort $w \in L$ mit $|w| \geq n$ aufgeteilt werden kann in $w = xyz$ mit

(1) $|xy| \leq n$,

(2) $y \neq \lambda$,

(3) $\forall i \in \mathbb{N} : xy^i z \in L$.

Sei $n \in \mathbb{N}$ beliebig. Betrachte das Wort $a^n b^n c^{n^2} \in L$ mit $|w| > n$. Betrachte eine beliebige Aufteilung

$$xyz = a^n b^n c^{n^2}$$

mit $|xy| \leq n$ und $|y| \geq 1$ (nach Bedingungen (1) und (2) des Pumpinglemmas). Daraus folgt:

- $xy = a^k$ mit $k \leq n$,
- $y = a^j$ mit $1 \leq j \leq k$ und $x = a^{k-j}$,
- $z = a^{n-k} b^n c^{n^2}$.

Da $n > 0$, ist bei wohlgeformten Wörtern die Anzahl an c's nicht unabhängig von der Anzahl an a's. Also bewirkt jede Veränderung an der Anzahl an a's (um $j \neq 0$), ohne dass die Anzahl an b's oder c's verändert wird, dass das entstehende Wort nicht in der Sprache sein kann (denn $(n \pm j) \cdot n \neq n^2$). Da y nur a's enthalten kann (wegen $|xy| \leq n$), gilt bspw. für $i = 0$, dass

$$xy^0 z = yz = a^{k-j} a^{n-k} b^n c^{n^2} = a^{n-j} b^n c^{n^2} \notin L$$

Wir sind zu einem Widerspruch gekommen und müssen die anfängliche Vermutung aufgeben, dass L vom Typ 3 ist.

(b) Skizzieren Sie kurz, wie Sie vorgehen würden, um zu zeigen, dass L vom Chomsky-Typ 1 ist.

/ 1

Lösung:

Es kann beispielsweise

- eine Typ-1-Grammatik,
- ein LBA,
- eine Turingmaschine mit linearer Bandbeschränkung angegeben werden oder
- ein nicht-kostruktiver Beweis geführt oder
- die Sprache in Polynomialzeit auf bspw. ein Problem $X \in P$ reduziert werden
- usw.

(c) Angenommen, das Negativergebnis aus (a) und das hypothetische positive Ergebnis aus (b) seien bekannt.

- In welchen Chomsky-Klassen könnte L dann liegen?
- In welchen liegt L gewiss?

/ 1

Lösung:

- Könnte: Typen 0, 1, 2.
- Gewiss: Typen 0, 1.

Aufgabe 3 **10 Punkte**

2015-H-03 **Grammatiken**

/ 10

Gegeben sei eine kontextfreie Grammatik durch

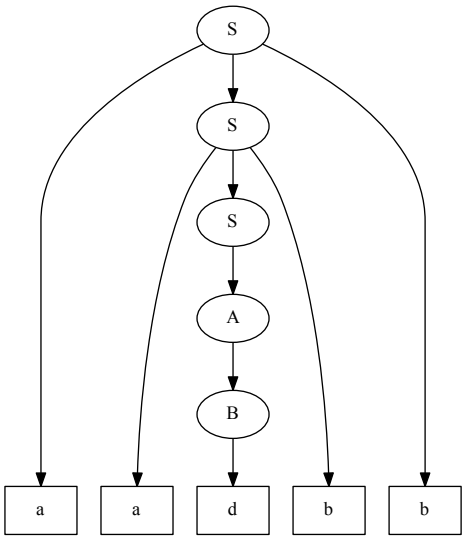
$$G = (\{A, B, E, F, S\}, \{a, b, c, d, e, f\}, P, S) \qquad \text{und}$$
$$\begin{aligned} P = \{S &\to \lambda \mid A \mid EF \mid aSb, \\ A &\to c \mid B, \\ B &\to d, \\ E &\to e \mid d \mid FF, \\ F &\to f \mid SA\} \end{aligned}$$

(a) Geben Sie eine vollständige Ableitungsfolge **oder** einen Ableitungsbaum für das Wort $\boxed{aadbb} \in L(G)$ an.

/ 1

Lösung:

Ableitungsbaum:



Ableitungsfolge: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaAbb \Rightarrow aaBbb \Rightarrow aadbb$

(b) Wandeln Sie G mit dem Algorithmus aus der Vorlesung in eine äquivalente Grammatik G_{CNF} in Chomsky-Normalform (CNF) um. Geben Sie für jeden der vier Schritte explizit an, welche Produktionen hinzukommen oder wegfallen. (Sie müssen nicht für jeden Schritt die vollständige Definition der Teilgrammatiken angeben.)

/ 4

Lösung:

*** Lambda-freie Grammatik ($S' \to \lambda$ wird am Ende hinzugefügt) ***

$$G_{(1)} = (\{A, B, E, F, S\}, \{a, b, c, d, e, f\}, P_{(1)}, S)$$

$$\begin{aligned} P_{(1)} = \{&A \to B \mid c, \\ &B \to d, \\ &E \to FF \mid e \mid d, \\ &F \to A \mid f \mid SA, \\ &S \to EF \mid aSb \mid ab \mid A\} \end{aligned}$$

*** Grammatik ohne einfache Umbenennungen ***

$$G_{(2)} = (\{A, E, F, S\}, \{a, b, c, d, e, f\}, P_{(2)}, S)$$

$$\begin{aligned} P_{(2)} = \{&A \to d \mid c, \\ &E \to FF \mid e \mid d, \\ &F \to f \mid d \mid c \mid SA, \\ &S \to EF \mid c \mid aSb \mid ab \mid d\} \end{aligned}$$

*** Grammatik mit isolierten Terminalen ***

$$G_{(3)} = (\{A, C_{(a)}, C_{(b)}, E, F, S\}, \{a, b, c, d, e, f\}, P_{(3)}, S)$$

$$\begin{aligned} P_{(3)} = \{&A \to d \mid c, \\ &C_{(a)} \to a, \\ &C_{(b)} \to b, \\ &E \to FF \mid e \mid d, \\ &F \to f \mid d \mid c \mid SA, \\ &S \to EF \mid c \mid C_{(a)}C_{(b)} \mid d \mid C_{(a)}SC_{(b)}\} \end{aligned}$$

*** CNF-Grammatik (inklusive $S' \to \lambda$) ***

$$G_{CNF} = (\{A, C_{(a)}, C_{(b)}, D_{(0)}, E, F, S, S'\}, \{a, b, c, d, e, f\}, P_{CNF}, S')$$

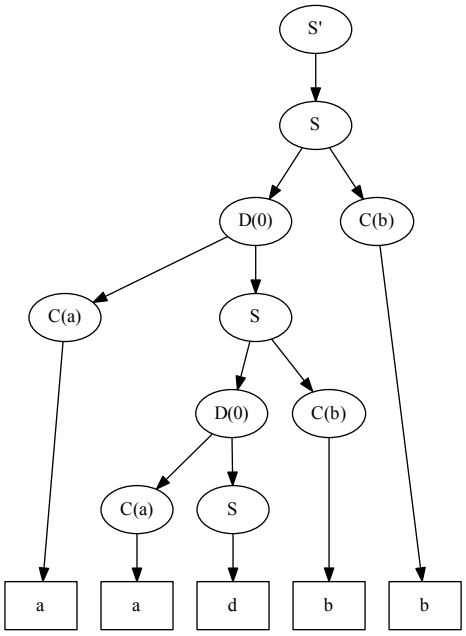
$$
\begin{aligned}
P_{CNF} = \{&A \to d \mid c,\\
&C_{(a)} \to a,\\
&C_{(b)} \to b,\\
&D_{(0)} \to C_{(a)}S,\\
&E \to FF \mid e \mid d,\\
&F \to f \mid d \mid c \mid SA,\\
&S \to EF \mid D_{(0)}C_{(b)} \mid c \mid C_{(a)}C_{(b)} \mid d,\\
&S' \to \lambda \mid S\}
\end{aligned}
$$

(c) Leiten Sie das Wort $\boxed{aadbb}$ bezüglich G_{CNF} (aus Teil b) ab.

/ 1

Lösung:

Ableitungsbaum:



Ableitungsfolge: $S' \Rightarrow S \Rightarrow D_{(0)}C_{(b)} \Rightarrow C_{(a)}SC_{(b)} \Rightarrow aSC_{(b)} \Rightarrow aD_{(0)}C_{(b)}C_{(b)} \Rightarrow aC_{(a)}SC_{(b)}C_{(b)} \Rightarrow aaSC_{(b)}C_{(b)} \Rightarrow aadC_{(b)}C_{(b)} \Rightarrow aadbC_{(b)} \Rightarrow aadbb$

(d) Gegeben sei eine zu G äquivalente Grammatik in Greibach-Normalform (GNF) durch

$$G_{GNF} = (\{A_1, A_4, A_5, A_7, B_7, S'\}, \{a, b, c, d, e, f\}, P_{GNF}, S')$$

und

$$
\begin{aligned}
P_{GNF} = \{S' \to & \lambda \mid A_4,\\
A_1 \to & c \mid d,\\
A_4 \to & c \mid d \mid aA_4A_5A_1A_7A_7 \mid dB_7A_7A_7 \mid aA_4A_5A_1B_7A_7A_7 \mid aA_5A_1A_7A_7 \mid dA_1A_7A_7\\
& \mid aA_4A_5 \mid cA_7A_7 \mid eA_7A_1A_7A_7 \mid dA_7 \mid eA_7A_1B_7A_7A_7 \mid fB_7A_7A_7\\
& \mid aA_5A_1B_7A_7A_7 \mid dA_7A_1A_7A_7 \mid cA_1B_7A_7A_7 \mid aA_5 \mid dA_1B_7A_7A_7 \mid eA_7\\
& \mid dA_7A_7 \mid dA_7A_1B_7A_7A_7 \mid cB_7A_7A_7 \mid fA_7A_7 \mid cA_1A_7A_7,\\
A_5 \to & b,\\
A_7 \to & c \mid d \mid f \mid dA_1B_7 \mid aA_4A_5A_1 \mid eA_7A_1B_7 \mid aA_5A_1B_7 \mid fB_7 \mid aA_5A_1 \mid eA_7A_1\\
& \mid dA_7A_1B_7 \mid cA_1B_7 \mid cA_1 \mid dB_7 \mid aA_4A_5A_1B_7 \mid dA_7A_1 \mid cB_7 \mid dA_1,\\
B_7 \to & aA_4A_5A_1A_7A_1 \mid cA_1A_7A_1 \mid dB_7A_7A_1 \mid aA_4A_5A_1B_7A_7A_1 \mid dA_7A_1A_7A_1B_7\\
& \mid aA_4A_5A_1B_7A_7A_1B_7 \mid aA_5A_1B_7A_7A_1B_7 \mid eA_7A_1A_7A_1 \mid cA_1B_7A_7A_1B_7\\
& \mid cB_7A_7A_1 \mid aA_5A_1A_7A_1B_7 \mid dA_7A_1B_7 \mid dA_7A_1B_7A_7A_1 \mid cB_7A_7A_1B_7\\
& \mid aA_4A_5A_1A_7A_1B_7 \mid dB_7A_7A_1B_7 \mid dA_7A_1A_7A_1 \mid dA_1A_7A_1B_7 \mid cA_7A_1\\
& \mid eA_7A_1A_7A_1B_7 \mid aA_5A_1B_7A_7A_1 \mid fB_7A_7A_1 \mid cA_1A_7A_1B_7 \mid cA_7A_1B_7\\
& \mid dA_7A_1B_7A_7A_1B_7 \mid fB_7A_7A_1B_7 \mid dA_1A_7A_1 \mid dA_1B_7A_7A_1 \mid fA_7A_1B_7\\
& \mid cA_1B_7A_7A_1 \mid eA_7A_1B_7A_7A_1B_7 \mid aA_5A_1A_7A_1 \mid dA_1B_7A_7A_1B_7 \mid fA_7A_1\\
& \mid dA_7A_1 \mid eA_7A_1B_7A_7A_1\}
\end{aligned}
$$

/ 4

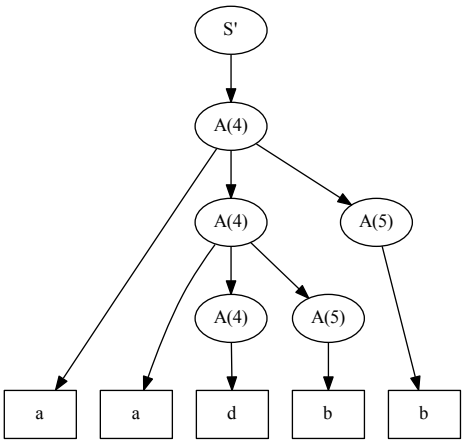
- Wie sind die Produktionen von GNF-Grammatiken allgemein definiert?
- Leiten Sie $\boxed{aadbb}$ bezüglich G_{GNF} ab.

  Hinweis: Sie benötigen nur die Nonterminale S', A_4 und A_5.
- Wie viele Ableitungsschritte werden **allgemein** bei GNF-Grammatiken benötigt, um ein Wort der Länge n abzuleiten? Wie viele sind es bei CNF-Grammatiken?

Lösung:

- Allgemeine Definition von GNF-Regeln: $N \times (T \times N^\star)$.

- Ableitungsbaum:



  Ableitungsfolge: $S' \Rightarrow A_4 \Rightarrow aA_4A_5 \Rightarrow aaA_4A_5A_5 \Rightarrow aadA_5A_5 \Rightarrow aadbA_5 \Rightarrow aadbb$.

- GNF-Grammatiken leiten Wörter der Länge n immer in n (oder $n+1$, falls $S' \rightarrow \ldots$ vorhanden) Schritten ab. Entsprechend wird *aadbb* hier in 6 Schritten abgeleitet. CNF-Grammatiken benötigen $2n - 1$ (oder $2n$; das „-1“ kommt daher, dass mit S oder S' schon vor der ersten Ableitung ein Nonterminal vorhanden ist, dazu kommt in jedem Schritt ein weiteres oder es wird ein Nonterminal in ein Terminal umgewandelt) Schritte, hier entsprechend 10 Schritte für *aadbb*.

Aufgabe 4 **8 Punkte**

2015-H-04 **Turingmaschine**

/ 8

Gegeben sei die Sprache

$$L = \left\{ w = b_1 \dots b_n \in \{0,1\}^n \;\middle|\; \sum_{i=1}^{n} u(b_i) = \sum_{k=1}^{n} g(b_k) \right\} \text{ und}$$

$$u(b_i) = \begin{cases} b_i & \text{falls } i \text{ ungerade,} \\ 0 & \text{sonst.} \end{cases}$$

$$g(b_k) = \begin{cases} b_k & \text{falls } k \text{ gerade,} \\ 0 & \text{sonst.} \end{cases}$$

L enthält somit alle Wörter, für die die Anzahl der Einsen an den geraden Stellen gleich der Anzahl der Einsen an den ungeraden Stellen ist. Es gilt also

- $\lambda, 0, 11, 011, 0110, 1001, 010100101 \in L$,
- $1, 01, 10, 101, 111, 1100001 \notin L$.

Geben Sie eine Turingmaschine $T = (E, B, S, \delta, s_0, F)$ an, für die gilt $L(T) = L$. T ist also ein Akzeptor, der in einem Endzustand anhält, falls $w \in L$. Geben Sie T vollständig an.

Hinweise:

- T darf die Bandinschrift w beliebig überschreiben.
- Der Schreib-/Lesekopf steht zu Beginn über dem linkesten Zeichen der Eingabe und terminiert, falls T akzeptiert, über dem $\star$-Feld rechts neben dem rechtesten beschriebenen Feld.
- Die Spaltenanzahl der vorgegebenen Tabelle (nächste Seite) entspricht nicht der minimal erforderlichen Anzahl an Bandsymbolen und die Zeilenanzahl nicht der minimal erforderlichen Anzahl an Zuständen.

Lösung:

$T = (\{0,1\}, \{0,1,\star\}, \{s_0, \ldots, s_3, s_e\}, \delta, s_0, \{s_e\})$

δ	0	1	$\star$
s_0	$(s_0, 0, R)$	$(s_1, 0, R)$	$(s_e, \star, N)$
s_1	$(s_2, 0, R)$	$(s_3, 0, L)$	
s_2	$(s_1, 0, R)$	$(s_1, 1, R)$	
s_3	$(s_3, 0, L)$	$(s_3, 1, L)$	$(s_0, \star, R)$
s_e			

Aufgabe 5	**8 Punkte**
2015-H-05	**Berechenbarkeit**

/ 8

Für eine Turingmaschine T sei $s(T)$ die Anzahl an Zustandsübergängen, die T **bei Eingabe des leeren Wortes** bis zum Halten durchführt. Es gelte $s(T) =_{def} 0$, falls die Turingmaschine nicht anhält.

Beispiel: Läuft eine Turingmschine T zwei Schritte nach rechts und hält dann, gilt $s(T) = 2$. Welche Bandsymbole sie dabei schreibt, ist irrelevant.

Gegeben sei weiter durch $\mathcal{T}_n$ für $n \in \mathbb{N}$ die Menge aller Turingmaschinen mit n Zuständen über dem Bandalphabet $\{1, \star\}$. Dann sei

$$S(n) =_{def} \max(\{s(T) \mid T \in \mathcal{T}_n\})$$

die maximale Anzahl an Zustandsübergängen $s(T)$, die irgendeine solche **nicht endlos laufende** Turingmaschine T mit n Zuständen durchführt.

(a) Geben Sie $S(1)$ an und begründen Sie Ihre Wahl.

Hinweis: Denken Sie daran, dass Sie nur die Bandsymbole 1 und $\star$ nutzen dürfen.

/ 2

Lösung:

Der eine Zustand kann einen Stern in eine Eins umwandeln und den Kopf nicht weiter bewegen; ist kein Übergang für das Bandsymbol 1 definiert, hält die Turingmaschine danach an, also ist $S(1) \geq 1$. Bewegt sich der Kopf in irgendeine Richtung, steht er danach wieder auf einem Stern, was den Anfangszustand wiederherstellt und zu einer Endlosschleife führt. Also ist $S(1) = 1$.

(b) Wie müsste man (als Mensch) vorgehen, um für ein festes $k \in \mathbb{N}$ den Wert $S(k)$ zu bestimmen?

/ 3

Lösung:

Es müsste für jede mögliche Turingmaschine mit k Zuständen (deren Anzahl exponentiell mit k steigt) gezeigt werden, ob sie anhält oder nicht, was im Allgemeinen ein nicht berechenbares Problem ist. Für jede feste Turingmaschine kann das jedoch (oft mit großem Aufwand) möglich sein – trotz Unberechenbarkeit des Halteproblems. Für alle (endlich vielen) Turingmaschinen, die anhalten, könnte man eine universelle Turingmaschine nutzen, um sie so lange zu simulieren, bis sie anhalten (was extrem lange dauern kann); zählt man dabei die Rechenschritte, ist $S(k)$ die größte Anzahl dieser Schritte bei irgendeiner der haltenden Turingmaschinen. Die nicht haltenden Turingmaschinen kann man ignorieren, weil für sie $s(T) = 0$ gilt. Obwohl das theoretisch ein anwendbares Verfahren ist, ist es in der Praxis schon für sehr kleine k nicht durchführbar (für alle $k \geq 4$ ist $S(k)$ bis heute unbekannt).

(c) Begründen Sie, dass $S(n)$ im Allgemeinen nicht berechenbar ist.

Hinweis: Das Halteproblem ist (auch bei leerer Eingabe) nicht berechenbar.

/ 3

Lösung:

Wäre die Funktion berechenbar, könnte man folgendermaßen das Halteproblem lösen: Sei T die Turingmaschine, für die getestet werden soll, ob sie auf dem leeren Wort hält. Zähle deren Zustände n und berechne $S(n)$. Simuliere T für $S(n)$ Schritte – hat sie bis dahin nicht angehalten, wird sie nie anhalten, da keine Turingmaschine mit n Zuständen mehr als $S(n)$ Schritte laufen kann, ohne in eine Endlossschleife zu geraten.

In der Tat wächst die Funktion noch schneller als die Busy Beaver-Funktion (vgl. Lösung der Aufgabe **2014-N-04** von letztem Jahr). Da ein Feld, um mit einer Eins beschrieben zu werden, erst besucht werden muss, wird pro Eins mindestens ein Konfigurationsübergang benötigt. Daher gilt offenbar: $S(n) \geq b(n)$.

Aufgabe 6 **9 Punkte**

2015-H-06 **CMOS und Schaltnetze**

/ 9

Die Funktion $f_n : \mathbb{B}^n \to \mathbb{B}$ sei bei Eingabe eines Binärstrings $B = b_1 b_2 \ldots b_{n-2} b_{n-1} b_n$ wie folgt definiert:

$$f_n(b_1, \ldots, b_n) = \left(\sum_{i=1}^{n} b_i \right) \text{ MOD } 2$$

f_n berechnet also den Rest bei Division durch 2 der Summe aller Einsen des Binärstrings B.

(a) Geben Sie eine CMOS-Schaltung an, die zwei Eingaben $b_1, b_2 \in \mathbb{B}$ erhält, diese als Binärstring $b_1 b_2 \in \mathbb{B}^2$ interpretiert und als Ausgabe den Wert $f_2(b_1 b_2)$ berechnet.

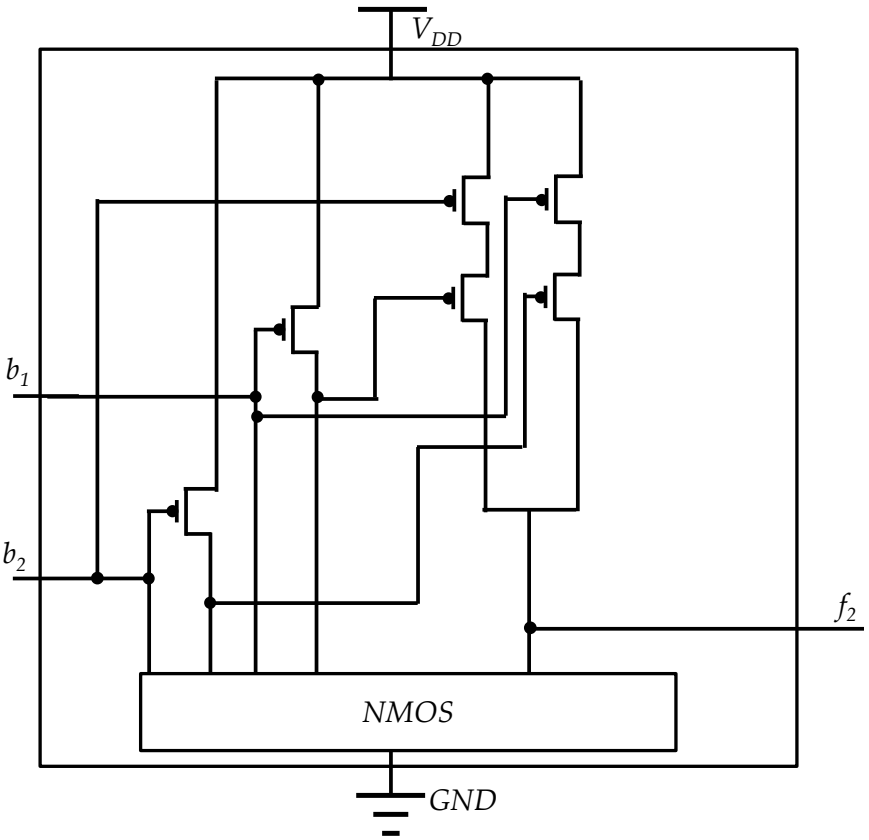
Hinweise:

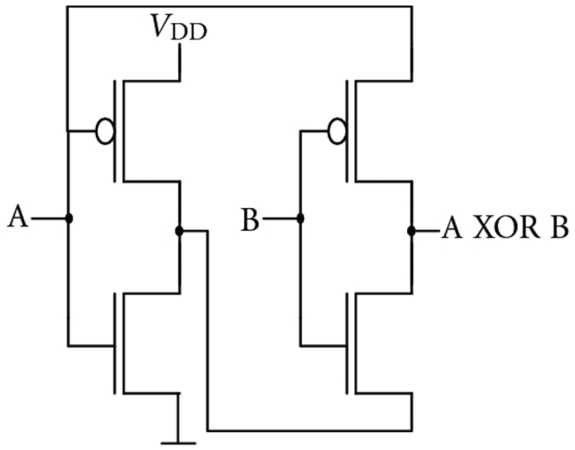
- Sie können sich zuerst überlegen, welchen Booleschen Ausdruck f_2 darstellt. Als Hilfsmittel können Sie die leere Tabelle verwenden, sie wird aber nicht gewertet.
- Zeichnen Sie nur den PMOS-Teil mit entsprechenden Verbindungen zur NMOS-Blackbox.

/ 7

Lösung:

b_1	b_2	f_2
0	0	0
0	1	1
1	0	1
1	1	0

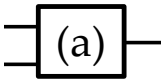




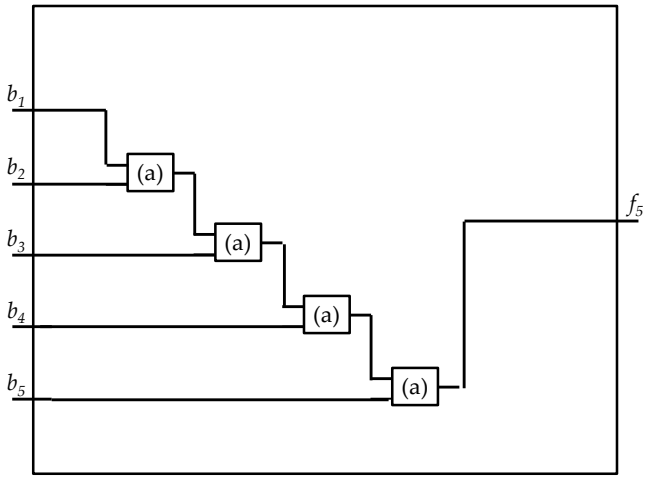
(b) Geben Sie ein Schaltnetz an, das als Eingabe einen Bitstring $b_1 \dots b_5$ erhält und als Ausgabe $f_5(b_1, \dots, b_5)$ berechnet. Verwenden Sie hierfür als Blackbox die Schaltung für f_2 aus Teilaufgabe (a) (siehe unten; Sie dürfen die Blackbox auch verwenden, wenn Sie (a) nicht gelöst haben).

/ 2

Blackbox ($f_2 \colon \mathbb{B}^2 \to \mathbb{B}$):



Lösung:



Bemerkung: Eine andere Schachtelung der Blackboxen ist auch möglich.

Aufgabe 7	**10 Punkte**
2015-H-07	**Zahlendarstellung und Kodierung**

/ 10

Gegeben seien die beiden Binärstrings $A, B \in \{0,1\}^8$ durch

$$A = 10010110$$

$$B = 01011000$$

(a) Interpretieren Sie A und B als Dualzahlen im 2er-Komplement. Addieren Sie A und B **ohne** sie vorher in Dezimalwerte umzuwandeln und geben Sie dann **das Ergebnis** im 2er-Komplement **und** als Dezimalwert an.

/ 2

Lösung:

$(10010110)_{2K,8}$ $[= -106_{10}]$
$(01011000)_{2K,8}$ $[= +88_{10}]$
———————————————
$(11101110)_{2K,8} = -18_{10}$

(b) Interpretieren Sie A und B als Gleitkommazahlen mit einem Vorzeichenbit, 4-Bit-Charakteristik und 3-Bit-Mantisse. Multiplizieren Sie A und B **im Gleitkommaformat**. Dokumentieren Sie Ihren Rechenweg in Stichworten und geben Sie dann **das Ergebnis** als Gleitkommazahl **und** Dezimalwert an.

/ 5

Lösung:

Allgemeine Darstellung einer Gleitkommazahl:

$$(-1)^{VZ} \cdot (1 + m') \cdot 2^{c-q}$$

Multiplikation durch

- Addition der Exponenten $e = c - q$:

$$\begin{aligned} e_A + e_B &= c_A - q + c_B - q = (0010)_2 - 7_{10} + (1011)_2 - 7_{10} \\ &= 2_{10} - 7_{10} + 11_{10} - 7_{10} = -1_{10} = e_{Produkt} \\ \Rightarrow c_{Produkt} &= -1_{10} + q = 6_{10} = (0110)_2 \end{aligned}$$

- und Multiplikation der Mantissen $m = 1 + m'$:

$$m_A \cdot m_B = (1 + 0.5 + 0.25) \cdot (1 + 0) = m_A$$

Daraus ergibt sich ein Produkt von $(10110110)_{GPZ} = -0.875_{10}$

(c) Interpretieren Sie A und B als die einzigen Codewörter eines Codes. Wie viele Bit-Fehler können maximal erkannt und wie viele maximal korrigiert werden?

/ 2

Lösung:

$h_c = 5 \Rightarrow$ 4-Fehler-erkennbar und 2-Fehler-korrigierbar.

(d) Geben Sie ein weiteres Codewort C an, so dass der resultierende Code mindestens 2-Fehler-erkennbar ist.

/ 1

Lösung:

C muss zu den beiden anderen Codewörtern mindestens einen Hammingabstand von 3 aufweisen:

10**01**011**0**	(A)
01**01**100**0**	(B)
$(\cdot\cdot\mathbf{10}\cdot\cdot\cdot\mathbf{1})_2$	(Neues Codewort C)

Aufgabe 8 **7 Punkte**

2015-H-08 **Huffman-Kodierung**

/ 7

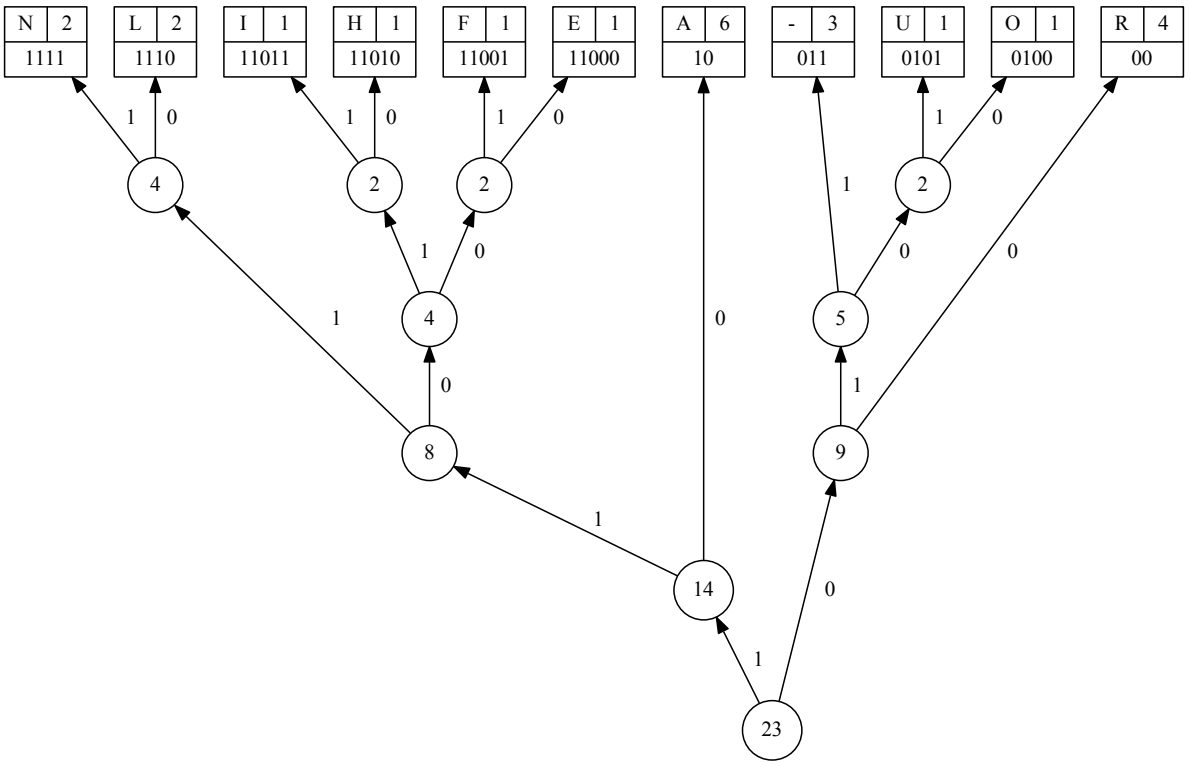
Folgende Zeichenkette der Länge 23 sei repräsentativ für Daten, die noch kommen sollen (– wird explizit kodiert):

NARRI-NARRO-ALAAF-HELAU

Erzeugen Sie anhand der durch die Zeichenkette gegebenen Häufigkeitsverteilung eine Huffman-Kodierung. Tragen Sie dazu die Häufigkeiten der Zeichen in die erste Tabelle ein, erstellen Sie einen entsprechenden Baum mit Angabe der Häufigkeiten an den Knoten und tragen Sie die Kodierung der Zeichen in die zweite Tabelle ein.

Zeichen	N	A	R	I	-	O	L	F	H	E	U	
Häufigkeit	2	6	4	1	3	1	2	1	1	1	1	

Lösung (Beispiel):



Aufgabe 9	**6 Punkte**
2015-H-09	**Von-Neumann-Architektur**

/ 6

Der Von-Neumann-Rechner enthält die wesentlichen Bestandteile

Eingabewerk, **Speicherwerk**, **Ausgabewerk**, **Rechenwerk** und **Steuerwerk**.

In diesem Zusammenhang treten die im Folgenden genannten Begriffe auf. Beschreiben Sie diese in kurzen Worten. Gehen Sie dabei insbesondere auf deren **Funktion** und deren **Zuordnung** zu den Bestandteilen des Von-Neumann-Rechners ein.

(a) Register

/ 1

Lösung:

Register dienen der **Speicherung von Operanden** und Resultaten von Operationen oder Zustandsbeschreibungen von Funktionseinheiten und sind im **Rechenwerk** zu finden.

(b) Central Processing Unit (CPU)

/ 2

Lösung:

Die Central Processing Unit (CPU) beschreibt die **Einheit des Rechenwerks mit dem Steuerwerk**. Dabei ist das Rechenwerk der funktionelle Kern eines Rechners, in dem **logische und arithmetische Operationen berechnet** sowie Vergleiche und logische Entscheidungen durchgeführt werden. Das Steuerwerk bestimmt die **Art und Reihenfolge der auszuführenden Operationen**.

(c) Maschinenbefehlszyklus

/ 2

Lösung:

Der Maschinenbefehlszyklus ist der Vorgang, den das **Steuerwerk** für jeden Befehl ausführt. Dieser untergliedert sich in **FETCH** (Befehl wird aus dem Arbeitsspeicher geladen), **DECODE** (Befehlsart wird erkannt und in Operations- und Operandenteil zerlegt) und **EXECUTE** (die Befehlsausführung wird initiiert, also alle an der Befehlsausführung beteiligten Funktionseinheiten werden mit den notwendigen Steuersignalen versorgt).

(d) Cache

/ 1

Lösung:

Der Cache ist ein Zwischenspeicher des **Speicherwerks**. Hier werden vor allem Daten abgelegt, die zu einem **aktuellen** Zeitpunkt vom Rechenwerk **zum Arbeiten benötigt werden**.

Aufgabe 10	**10 Punkte**
2015-H-10	**Assembler, Adressierungsarten**

/ 10

Die Befehle einer Assembler-Sprache seien folgendermaßen aufgebaut, wobei `Q` für Quelle steht und `Z` für Ziel:

```
OpCode Q1, (Q2,) Z
```

Für **unmittelbare Adressierung** wird das Präfix # verwendet, **direkte Adressierung** geben wir ohne Präfix an. Sprungbefehle sind unbedingt (`JUMP L`) oder bedingt (`JUMPZERO L`).

Zusätzlich verwenden wir den Befehl

```
STORE_IND Q, Z
```

um `Q` in einen von den Registern getrennten Hauptspeicherraum zu schreiben, wobei **indirekte Adressierung bezogen auf** `Z` verwendet wird. Adressen von Registern (im Rechenwerk) sind von der Form `Rk` ($k \in \mathbb{N}_0$), Adressen von Speicherzellen (im Hauptspeicher) sind natürliche Zahlen `k` ($k \in \mathbb{N}_0$).

Gegeben sei das folgende Assemblerprogramm.

```
            LOAD        R8 ACC      *lade Registerinhalt in Akkumulator*

outerloop   JUMPZERO    fertig      *springe zu Marke falls '0' in Akkumulator*
            STORE       R8 R1       *speichere Inhalt von R8 in R1*
            STORE       R8 R2
            STORE       #0 R3       *speichere Konstante '0' in R3*

innerloop   ADD         R2 R3 R3    *R3 = R2 + R3*
            SUBTRACT    R1 #1 R1    *R1 = R1 - 1*
            LOAD        R1 ACC
            JUMPZERO    store
            JUMP        innerloop   *springe zu Marke*

store       STORE_IND   R3 R2       *STORE mit indirekter Addressierung*
            SUBTRACT    R8 #1 R8
            LOAD        R8 ACC
            JUMP        outerloop

fertig      STORE_IND   R8 R8       *STORE mit indirekter Adressierung*
            HALT                    *Programmende*
```

(a) Welcher Wert wurde nach **dem ersten** Durchlauf der **„outerloop“** an welche Adresse **im Hauptspeicher** geschrieben, wenn zu Beginn der Rechnung

$$\mathrm{R8} = n \in \mathbb{N}$$

gilt?

/ 6

Lösung:

Nach einem Durchlauf der „outerloop“ ergibt sich n^2 an Hauptspeicheradresse n.

(b) Welche Werte stehen dann nach einem **vollständigen** Durchlauf des **ganzen** Programms an welchen Stellen **im Hauptspeicher**?

/ 3

Lösung:

Das Programm speichert an Hauptspeicheradresse x den Wert x^2 für $x = \{1, \ldots, 8\}$.

(c) Was würde passieren, wenn statt `STORE_IND` nur `STORE` verwendet würde?

Lösung:

Das Ergebnis würde nicht an die Hauptspeicheradresse geschrieben werden, sondern in Register $R2$. Dieses wird im nächsten Durchlauf gleich überschrieben, das Ergebnis wäre verloren.

/ 1

Aufgabe 11	**5 Punkte**
2015-H-11	**Betriebssysteme**

/ 5

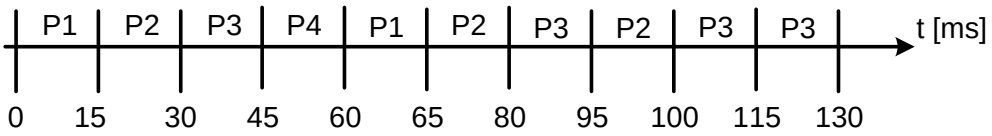
In der Warteschlange eines Prozessors befinden sich die folgenden Prozesse $P1$ bis $P4$, die in dieser Reihenfolge beim Prozessor ankommen:

Prozesse	CPU-Zeit in ms	Priorität
P1	20	1
P2	35	2
P3	60	1
P4	15	1

(a) Teilen Sie den Prozessen Rechenzeit gemäß dem Round Robin Verfahren zu. Die Zeitscheibe sei dabei in feste Zeitspannen der Länge $Z = 15$ms unterteilt. Veranschaulichen Sie Ihr Ergebnis auf dem gegebenen Zeitstrahl.

/ 2

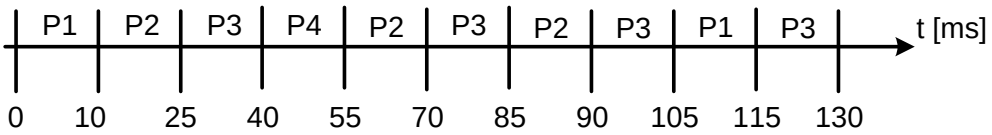
Lösung:



(b) Gehen Sie jetzt davon aus, dass $P1$, nachdem er 10ms bearbeitet wurde, auf das Ergebnis von $P2$ warten muss. Die Zeitspannen der Zeitscheibe betragen weiterhin $Z = 15$ms. Veranschaulichen Sie Ihr Ergebnis auf dem gegebenen Zeitstrahl.

/ 1

Lösung:



(c) Betrachten Sie nun ein prioritätenbasiertes Zuteilungsverfahren (2 ist dabei die höchste Prioritätsstufe). Nennen Sie zwei Möglichkeiten, die CPU-Zeit zwischen Prozessen gleicher Priorität aufzuteilen und geben Sie für beide Möglichkeiten die Reihenfolge an, in der die Prozesse $P1$ bis $P4$ abgearbeitet werden.

/ 2

Lösung:

- First Come, First Serve (FCFS): $P2$, $P1$, $P3$, $P4$
- Shortest Job First (SJF): $P2$, $P4$, $P1$, $P3$
- ...