Aufgabe 1. Endliche Automaten (10 Punkte)

/ 10

Gegeben seien die folgenden Sprachen L und ihr Komplement $\overline{L}$:

$$L = \left\{ w \in \{a,b\}^\star \mid w = a^n \prod_{i=1}^{k} (ba^{m_i}) = a^n ba^{m_1} \cdots ba^{m_k} \text{ mit } k \geq 0; m_i, n \geq 1 \right\},$$

$\overline{L} = \{a,b\}^\star \backslash L$, d.h. die Menge aller Wörter, die nicht in L enthalten sind.

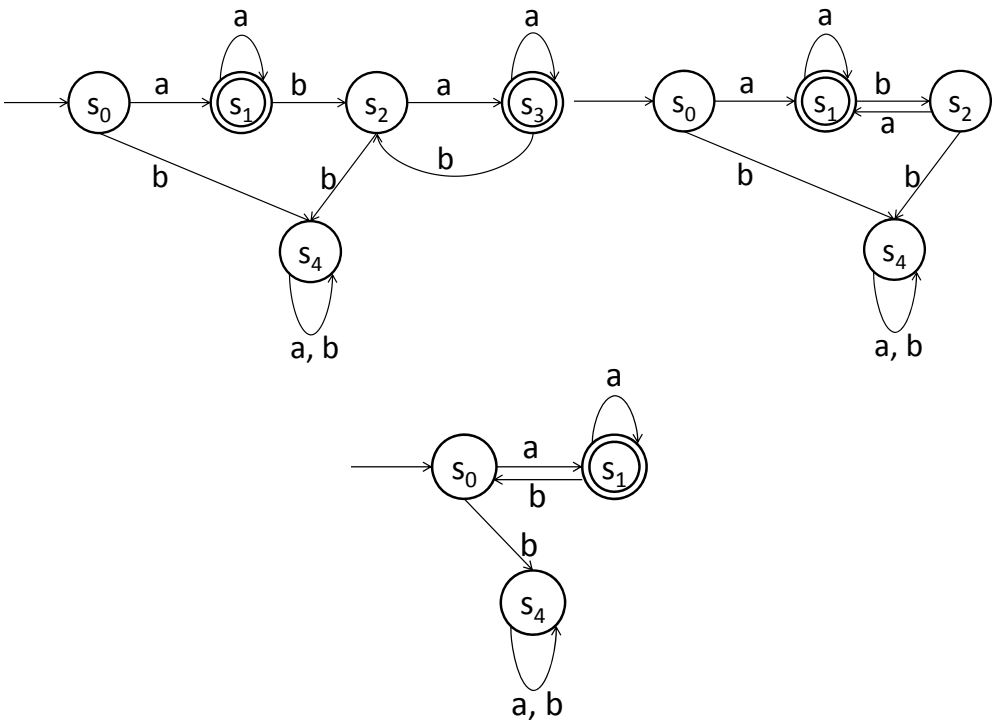
Es gilt beispielsweise:

$$a, aa, aababaaaaba, abababaaa, \in L$$
$$\lambda, bbb, abbb, babab, abababbaa \notin L.$$

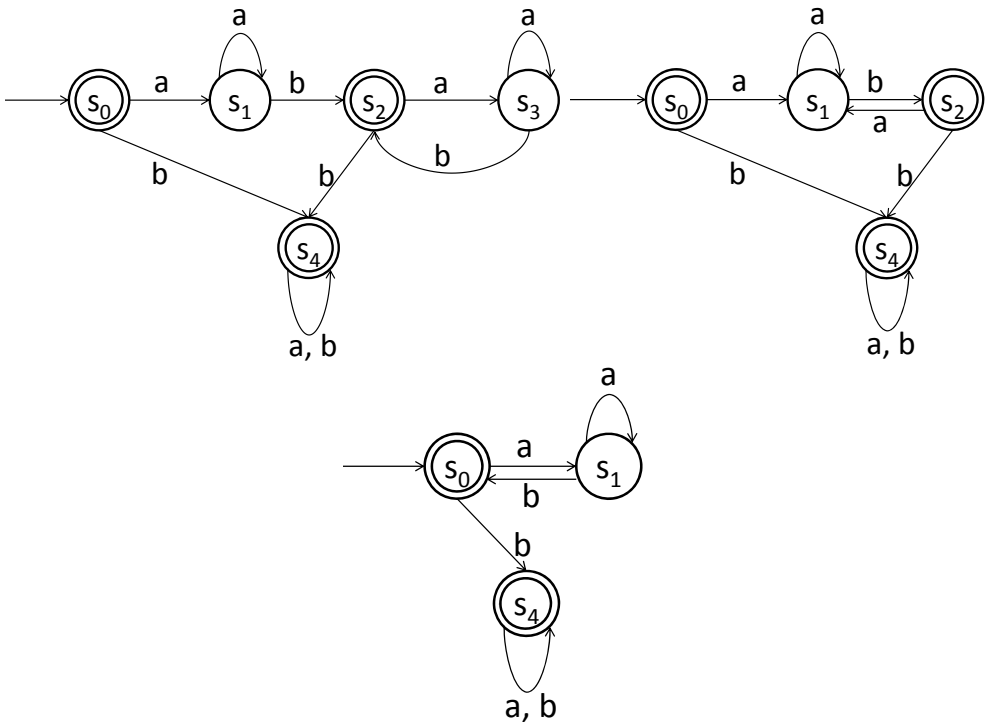
(a) Geben Sie einen deterministischen endlichen Automaten A an mit $L(A) = \overline{L}$. Geben Sie A vollständig an.

/ 6

Lösung:
Am einfachsten stellt man zuerst den Automaten für die Sprache L dar, und invertiert anschließend die Endzustände.
δ des Automaten zur Sprache L (drei Alternativen – Reduktion war nicht erforderlich):



Der invertierte Automat lautet:
$A = \{E, S, \delta, s_0, F\} = \{\{a, b\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta, s_0, \{s_0, s_2, s_4\}\}$
mit δ (drei Alternativen – s. o.):



(b) Geben Sie einen regulären Ausdruck α an mit $L(\alpha) = \overline{L}$.

/ 4

Lösung:

$$\alpha = \emptyset^\star + b(a+b)^\star + aa^\star b(aa^\star b)^\star(b(a+b)^\star + \emptyset^\star)$$
$$\text{oder } \alpha = \emptyset^\star + b(a+b)^\star + aa^\star b(b(a+b)^\star + (aa^\star b))^\star$$
$$\text{oder } \alpha = (aa^\star b)^\star(b(a+b)^\star + \emptyset^\star)$$

Aufgabe 2. Pumping-Lemma, Chomsky-Hierarchie (12 Punkte)

/ 12

Gegeben sei die Sprache

$$L = \{w \in \{0,1\}^\star \mid \exists v \in \{0,1\}^\star : w = vv\}.$$

Wörter aus L bestehen also aus der Hintereinanderschreibung zweier identischer Abschnitte. Es gilt beispielsweise:

$$\lambda, 00, 11, 0101, 101101 \in L;$$
$$01, 101, 110011, 1011001 \notin L.$$

(a) Zeigen Sie mit dem Pumping-Lemma (PPL) für Typ-3-Sprachen, dass L nicht vom Chomsky-Typ 3 ist.

/ 5

Lösung:
Angenommen, es gäbe einen endlichen Automaten A mit n Zuständen, sodass $L(A) = L$. Dann gibt es nach PPL für das Wort $w = 0^n10^n1 \in L$ (mit $|w| \geq n$) eine Zerlegung $w = xyz$, mit:

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

(2) $|y| > 0$ und

(3) $\forall i \in \mathbb{N}_0 : xy^iz \in L$.

Wegen (1) und (2) gilt $y = 0^j$ mit $0 < j \leq n$ und y enthält nur 0en aus dem ersten Teil des Wortes.

Wir betrachten das Wort $xy^0z = 0^{n-j}10^n1 \notin L$. Das Wort kann nicht in der Sprache liegen, weil $n - j \neq n$, und damit haben wir gezeigt, dass die o. a. Zerlegung entgegen der Annahme nicht existiert. Wir müssen also die ursprüngliche Annahme aufgeben, dass $L = L(A)$ für einen endlichen Automaten A.

(b) Argumentieren Sie in eigenen Worten unter Zuhilfenahme des PPLs für Typ-2-Sprachen, warum die Sprache L auch nicht vom Chomsky-Typ 2 sein kann.

Hinweis: Sie müssen das PPL nicht formal anwenden, es reicht die Beweisidee. Geben Sie aber insbesondere ein Pumpwort $w \in L$ in Abhängigkeit der PPL-Variablen n an und beschreiben Sie kurz, wie man pumpen könnte, um aus der Sprache zu fallen.

$$w = \underbrace{0^n}_{A}\underbrace{1^n}_{B}\underbrace{0^n}_{C}\underbrace{1^n}_{D} = uvwxy$$

/ 3

Lösung: Wegen der PPL-Regel „$|vwx| \leq n$“ kann vwx nicht mehr als zwei der vier Abschnitte A, B, C, D enthalten. Beim Pumpen mit $i = 0$ löscht man also nur aus maximal zwei Abschnitten Zeichen, während aus den anderen nichts gelöscht wird. Damit gilt $uv^iwx^iy \notin L$. Wir müssen also die Annahme aufgeben, dass $L = L(G)$ für eine kontextfreie Grammatik G. (Eine detaillierte Fallunterscheidung ist laut Aufgabenstellung nicht notwendig.)

(c) Wie könnte man beweisen, dass L vom Chomsky-Typ 1 ist? Geben Sie zwei der prinzipiellen Alternativen an, ohne den Beweis zu konkretisieren.

/ 2

Lösung:
Angabe einer Grammatik oder Angabe einer monotonen Grammatik oder Angabe einer linear beschränkten Turingmaschine.

(d) Kreuzen Sie an, mit welchen der folgenden Automatentypen bzw. Grammatiktypen man nach den Ergebnissen aus (a), (b) und (c) die Sprache L prinzipiell erkennen bzw. erzeugen könnte?

/ 2

Lösung:

- ◯ Endlicher Automat
- ◯ Deterministischer oder nichtdeterministischer Kellerautomat
- ⊗ Linear beschränkte Turingmaschine
- ⊗ Turingmaschine
- ◯ Rechtslineare Grammatik
- ◯ Kontextfreie Grammatik
- ⊗ Kontextsensitive oder monotone Grammatik
- ⊗ Allgemeine Grammatik

Hinweis: Sie erhalten 1/4 Punkte für jede richtige Antwort („Kreuz“ bzw. „kein Kreuz“) und $-1/4$ für jede falsche. Sie können nicht weniger als 0 Punkte erhalten.

Aufgabe 3. Kellerautomaten (9 Punkte)

/ 9

Gegeben sei für $E = \{0, 1\}$ folgende Sprache L:

$$L = \{w \in E^\star \mid w = 0^n 1^{2n+2} \text{ mit } n \geq 0\}.$$

Es gilt beispielsweise:

$$11, 01111, 00111111, 00011111111 \in L;$$
$$\lambda, 0, 0101, 000111, 11000000, 000011111111 \notin L.$$

(a) Geben Sie für L als Akzeptor einen deterministischen Kellerautomaten

$$KA = (E, S, K, \delta, s_0, k_0, F)$$

an. Geben Sie KA vollständig an.

/ 7

Lösung:
Da die Sprache kontextfrei ist, muss hier ein Kellerautomat angegeben werden:
$KA = (\{0, 1\}, \{s_0, s_1, s_2, s_e\}, \{k_0, 0, 1, a\}, \delta, s_0, k_0, \{s_e\})$

δ :

$\delta(s_0, 0, k_0) = (s_0, 0k_0)$
$\delta(s_0, 0, 0) = (s_0, 00)$
$\delta(s_0, 1, k_0) = (s_2, k_0)$
$\delta(s_0, 1, 0) = (s_1, a)$
$\delta(s_1, 1, a) = (s_1, \lambda)$
$\delta(s_1, 1, 0) = (s_1, a)$
$\delta(s_1, 1, k_0) = (s_2, k_0)$
$\delta(s_2, 1, k_0) = (s_e, k_0)$

(b) Geben Sie für den von Ihnen entworfenen Akzeptor die Erkennung des Testwortes 00111111 an.

/ 2

Lösung: Erkennung des Testwortes 00111111:

$(s_0, 00111111, k_0) \vdash (s_0, 0111111, 0k_0) \vdash (s_0, 111111, 00k_0) \vdash (s_1, 11111, a0k_0) \vdash$
$(s_1, 1111, 0k_0) \vdash (s_1, 111, ak_0) \vdash (s_1, 11, k_0) \vdash (s_2, 1, k_0) \vdash (s_e, \lambda, k_0)$

Aufgabe 4. Grammatiken (7 Punkte)

/ 7

Gegeben sei die Sprache

$$L = \{w \in \{a\}^\star \mid |w| = 2^n, n \geq 0\}.$$

Es gilt:

$$a, aa, aaaa, aaaaaaaa, \ldots \in L;$$
$$\lambda, aaa, aaaaa, aaaaaa, aaaaaaa, \ldots \notin L.$$

Geben Sie eine kontextsensitive, monotone oder allgemeine Grammatik $G = (N, T, P, S)$ an mit $L(G) = L$. Geben Sie die Grammatik vollständig an.

Hinweise:

- Erzeugen Sie zunächst einen Platzhalter A (Non-Terminal) für ein einzelnes a und markieren Sie Anfang und Ende des Wortes jeweils mit einem weiteren Non-Terminal.
- Realisieren Sie eine Verdopplung, indem Sie ein Non-Terminal von hinten nach vorne wandern lassen und bei jedem Passieren eines A dieses durch zwei A ersetzen.

Lösung:

$$G = (\{S, A, B, H, V\}, \{a\}, P, S)$$

monotone Grammatik:

$$\begin{aligned} P = \{&S \to VH \mid a, \\ &H \to BAH, \\ &AB \to BAA, \\ &VB \to VA, \\ &V \to a, \\ &A \to a, \\ &H \to a\} \end{aligned}$$

allgemeine Grammatik:

$$\begin{aligned} P = \{&S \to VAH \mid a, \\ &AH \to ABH, \\ &AB \to BAA, \\ &VB \to \lambda, \\ &H \to \lambda, \\ &A \to a\} \end{aligned}$$

Aufgabe 5. Komplexitäts- und Berechenbarkeitstheorie (7 Punkte)

/ 7

Gegeben seien die aus der Vorlesung bekannten Komplexitätsklassen

$$P, NP, NP\text{-schwer und } NP\text{-vollständig.}$$

(a) Geben Sie für jede der angegebenen Komplexitätsklassen eine Definition an.

/ 4

P: Menge aller Probleme, die in deterministisch polynomieller Zeit lösbar sind.

NP: Menge aller Probleme, die in nichtdeterministisch polynomieller Zeit lösbar sind.

NP-schwer: Menge aller Probleme Q für die für alle Probleme $Q' \in NP$ gilt: $Q' \leq_{pol} Q$.

NP-vollständig: Teilmenge von NP-schwer, die in NP liegt.

(b) Gegeben sei das folgende Optimierungsproblem auf Platinen:

Für eine Platine mit n Transistoren, die auf ihrer Oberfläche angebracht sind, sollen alle Transistoren durch eine Leitung, die ebenfalls auf der Oberfläche der Platine angebracht wird, miteinander verbunden werden, wobei jeder Transistor genau einmal von der Leitung durchkreuzt werden soll.

Gesucht ist ein Leitungsverlauf, bei dem die Länge der Leitung minimal ist.

(Gehen Sie davon aus, dass es für Kreuzungspunkte der Leitung bereits eine Lösung gibt; ihre Anzahl muss also nicht optimiert werden.)

Welche Komplexität wird Ihr Algorithmus in Abhängigkeit von n vermutlich mindestens haben? Wovon hängt das ab? Begründen Sie kurz.

/ 3

Lösung:

Der Algorithmus ist im Wesentlichen TSP, also NP-vollständig . Vermutlich benötigt der Algorithmus mindestens exponentiellen Zeitaufwand, also $\Omega(k^n)$ für eine Konstante k . Wenn allerdings $P = NP$ gelten sollte, ist der Zeitaufwand polynomiell, also $O(n^k)$ für eine Konstante k . Das liegt daran, dass im Fall $P = NP$ alle NP-vollständigen Probleme in P liegen .

Aufgabe 6. BDDs und Schaltnetze (10 Punkte)

/ 10

Gegeben sei die folgende Wahrheitstabelle, die eine Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ über den Booleschen Variablen a, b, c definiert:

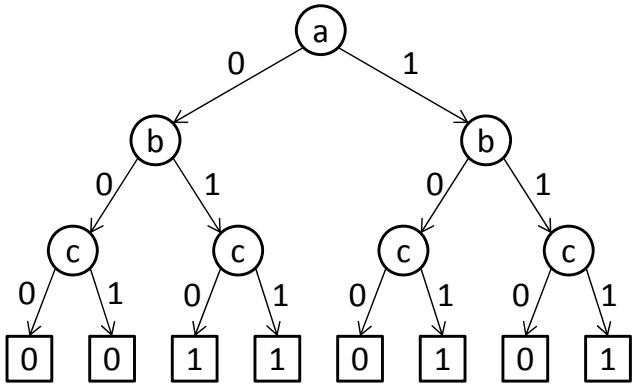
a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

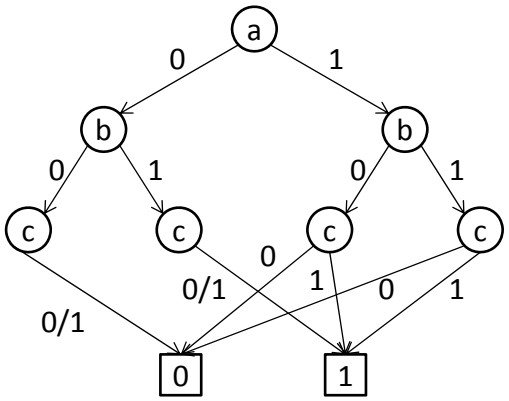
(a) Geben Sie ein Binary Decision Diagram (BDD) für die Funktion f an. Gehen Sie von der folgenden Variablenreihenfolge aus:

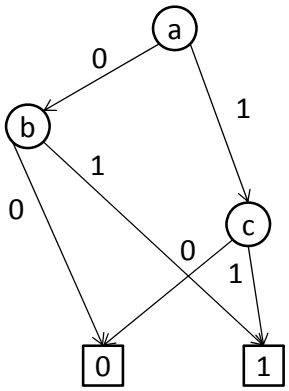
$$a \to b \to c.$$

/ 5

Lösung:



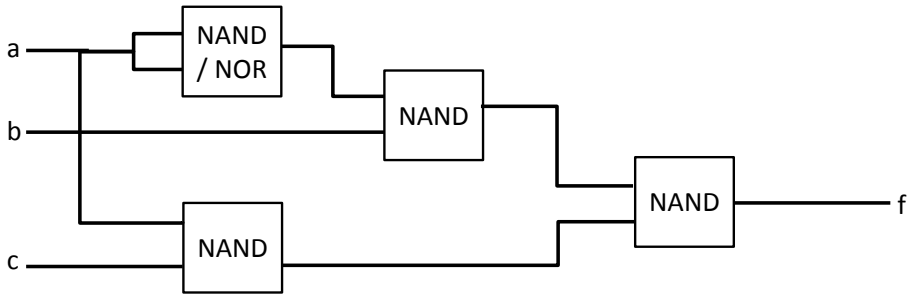




(b) Geben Sie ein Schaltnetz an, das ausschließlich aus den Bausteinen *NAND* und *NOR* besteht und die Funktion f berechnet.

/ 5

Lösung:



Aufgabe 7. Kodierung (11 Punkte)

/ 11

(a) Kann folgende Kodierung bestehend aus den Codewörtern a, b, c, d durch den Huffman-Algorithmus entstanden sein? Begründen Sie kurz.

/ 2

a	0010
b	0100
c	00
d	011

Lösung:
Diese Kodierung kann nicht durch den Huffman-Algorithmus entstanden sein, da die Kodierung die Fano-Bedingung nicht erfüllt. Die Kodierungen, die von dem Huffman-Algorithmus erzeugt werden, erfüllen aber die Fano-Bedingung.

(b) Für eine Kodierung stehen bis zu 4 Bits zur Verfügung. Wie viele Coderwörter können damit maximal kodiert werden? Wie viele Codewörter können damit maximal kodiert werden, wenn die Fano-Bedingung eingehalten werden soll?

/ 2

Lösung:
Anzahl der Codewörter ohne Fano-Bedingung: $n = 2^1+2^2+2^3+2^4 = 2+4+8+16 = 30$
Anzahl der Codewörter mit Fano Bedingung: $n = 2^4 = 16$

(c) Gegeben sei folgende Kodierung von e, f, g, h:

/ 7

e	00000
f	10011
g	11100
h	01111

(1) Wie viele gleichzeitige Bitfehler können beim Empfang eines verfälschten Codewortes im schlechtesten Fall maximal erkannt werden?

Lösung:
Hierzu muss die Fehlererkennung berechnet werden:
Hammingzahl $h_c = 3$
$\rightarrow$ Fehlererkennbarkeit $k = h_c - 1 = 3 - 1 = 2$

(2) Wie viele gleichzeitige Bitfehler können beim Empfang eines verfälschten Codewortes im schlechtesten Fall maximal korrigiert werden?

Lösung:
Hierzu muss die Fehlerkorrigierbarkeit berechnet werden:
Hammingzahl $h_c = 3$
$\rightarrow$ Fehlerkorrigierbarkeit $k = (h_c - 1)/2 = (3 - 1)/2 = 1$

(3) Welches Codewort / welche Codewörter sollten mit größter Wahrscheinlichkeit übermitteln werden, wenn Sie die Wörter 11011 und 11010 empfangen?

Lösung:

i. $11011 \rightarrow 10011(f)$, da hierzu nur eine $h_c = 1$

ii. $11010 \rightarrow 10011(f$ oder $g)$ oder 11100, da hierzu beide eine $h_c = 2$ als minimalen Wert besitzen

(4) Machen Sie den Code für e, f, g, h sicherer, indem Sie ein weiteres Bit anhängen. Geben Sie dieses weitere Bit in der folgenden Tabelle an und begründen Sie, wie der Code dadurch sicherer wird.

Lösung:

		gerade	ungerade
e	00000	0	1
f	10011	1	0
g	11100	1	0
h	01111	0	1

Durch das Anhängen des Prüfbits (Parity-Bit) wird die Anzahl der 1 auf ‘gerade‘/ ‘ungerade‘ gesetzt. Dadurch wird die Hammingzahl h_c um 1 erhöht was wiederum die Fehlererkennbarkeit erhöht.

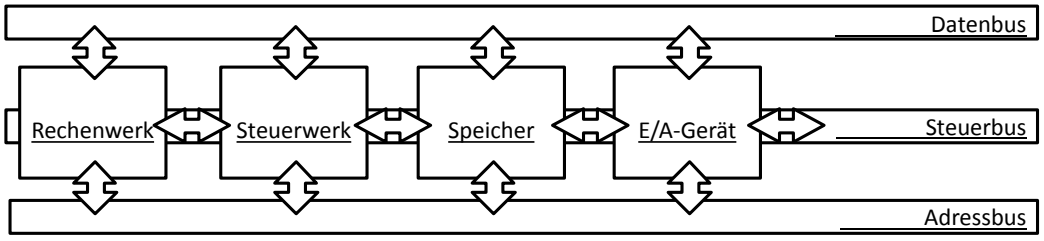
Aufgabe 8. Von Neumann-Rechner (7 Punkte)

/ 7

(a) In einem von Neumann-Rechner erfolgt die Übertragung von Informationen über verschiedene Busse. Benennen Sie in der folgenden Abbildung die verschiedenen Busse sowie die jeweiligen Funktionseinheiten des von Neumann-Rechners.

/ 3

Lösung:



(b) Ein Bus soll 400 MB an Daten übertragen. Der Bus hat eine Breite von 32 Bit und eine Taktfrequenz von 10 MHz. Wie lange dauert die Übertragung der Daten?

/ 2

Lösung:
Bandbreite = $32bit \cdot 10MHz = 4Byte \cdot 10.000.000/sec = 40MB/sec$
$\rightarrow$ Übertragungsdauer = $400MB/40MB/sec = 10sec$

(c) Erklären Sie in kurzen Worten den Begriff „Daisy Chain“.

/ 2

Lösung:
Daisy-Chain ist eime dezentrale Steuerung der Buszuteilung. Die Sendewünsche werden über eine gemeinsame Meldeleitung verarbeitet, wobei der erste sendewillige Teilnehmer das Verfügbarkeitssignal erhält.

Aufgabe 9. Assembler (10 Punkte)

/ 10

Die Befehle der in der Vorlesung vorgestellten Assembler-Sprache sind folgendermaßen aufgebaut, wobei 'Q' für Quelle steht und 'Z' für Ziel:

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

Für unmittelbare Adressierung wird das Präfix '#' verwendet. Ein bedingter Sprungbefehl ist *JNZ* (JumpNotZero), der den Befehlszähler genau dann zu Label L springen lässt, falls $Q \neq 0$. Die Notation des Sprungbefehls ist:

```
JNZ Q L
```

Gegeben sei das folgende Assemblerprogramm.

```
1.      STORE R1, R2
2.      STORE R1, R3
3. L1: MULTIPLY R1, R2, R1      [3. L1: STORE R1 R2 - für Teil (c)]
4.      ADD R2, R1, R1
5.      SUBTRACT R3, #1, R3
6.      JNZ R3, L1
```

(a) Am Anfang der Rechnung stehe im Register $R1$ der Wert $R1^A = n \in \mathbb{N}^+$. Wie hängt der Endwert $R1^E := f(n)$ des Registers $R1$ am Ende der Rechnung von n ab, welche Funktion $f(n)$ berechnet das Programm also?

/ 6

Lösung: Es wird für $n > 0$ die Funktion

$$f(n) = \underbrace{((((\ldots) \cdot n) + n) \cdot n + n) \cdot n + n}_{n \text{ mal}} = \sum_{i=1}^{n+1} n^i$$

berechnet.

(b) Was geschieht, wenn zu Beginn der Wert $R1^A = n = 0$ in $R1$ steht?

/ 1

Lösung:
für $n = 0$ wird die Funktion nicht korrekt berechnet, es kommt nämlich zu einer Endlosschleife, da $R3$ in Zeile 6 nie 0 ist.

(c) Welche Funktion $f'(n)$ wird berechnet, wenn man den Befehl „MULTIPLY R1, R2, R1" in Zeile 3 durch „STORE R1, R2" ersetzt (es gelte wieder $n \in \mathbb{N}^+$)?

/ 3

Lösung:
$f'(n) = 2^n n$

Aufgabe 10. Dateiorganisation (7 Punkte)

/ 7

Eine Möglichkeit der Organisation von Dateien ist die sogenannte Hash-Organisation. Im folgenden seinen zwei verschiedene Hash-Funktionen $h : \mathbb{N} \to \mathbb{N}$ mit dem Primärschlüssel s gegeben:

$$h_1(s) = s \bmod 13$$
$$h_2(s) = s \bmod 8$$

(a) Welche Hash-Funktion würden Sie bevorzugen? Begründen Sie kurz.

/ 2

Lösung:
Wahl der Hash-Funktion $h_1(s)$, da der Divisor eine Primzahl ist und diese für eine gleichmäßige Verteilung der Werte sorgt.

(b) Legen Sie im folgenden die Hashfunktion $h_2(s)$ zugrunde. Speichern Sie folgende Werte in der nachstehenden Datenbank. Verwenden Sie hierbei zur Kollisionsbehandlung lineares Austesten auf Satzebene.

/ 5

$$10, 30, 25, 68, 49, 11, 34$$

Lösung:

relative Satznummer	Satz mit Schlüssel
0	
1	25 (49 wird in 3 gespeichert)
2	10 (34 wird in 7 gespeichert)
3	49 (11 wird in 5 gespeichert)
4	68
5	11
6	30
7	34