Karlsruher Institut für Technologie

Institut für Angewandte Informatik
und Formale Beschreibungsverfahren

14.02.2017

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

Name: _____ Vorname: _____

Matr.-Nr.: _____ Semester: _____ (WS 2016/17)

Ich bestätige, dass ich die folgenden Angaben gelesen und mich von der Vollständigkeit dieses Klausurexemplars überzeugt habe (Seiten 1-24).

_____

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
(8)	(12)	(9)	(8)	(8)	(6)	(6)	(10)	(8)	(8)	(7)						(90)

Aufgabenübersicht

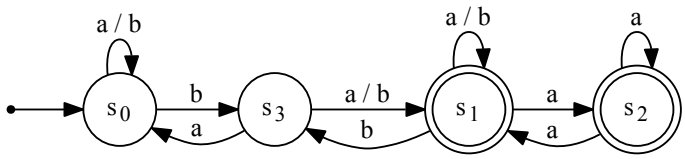
Aufgabe 1	**8 Punkte**
2017-H-01	**Endliche Automaten**

/ 8

Gegeben sei der folgende **nichtdeterministische** endliche Automat:

$$A = (\{a, b\}, \{s_0, \ldots, s_3\}, \delta, s_0, \{s_1, s_2\})$$

δ:



Erstellen Sie mithilfe des aus der Vorlesung bekannten Algorithmus einen **deterministischen** endlichen Automaten $A' = (E', S', \delta', s'_0, F')$ mit $L(A') = L(A)$ und geben Sie diesen vollständig an.

Hinweis: Geben Sie insbesondere ein Zustandsüberführungsdiagramm an. Nutzen Sie die vorgegebene Tabelle.

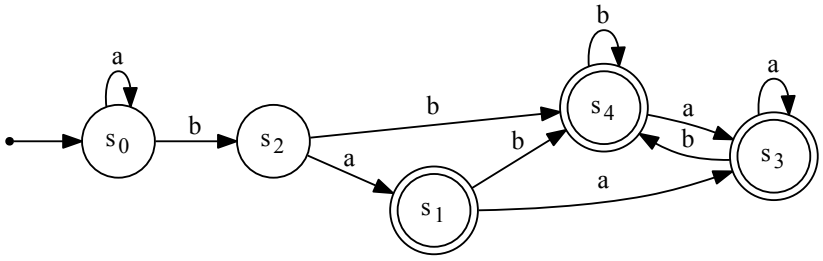
Lösung:

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

Hieraus ergibt sich der Automat:

$$A' = (\{a, b\}, \{s_0, \ldots, s_5\}, \delta', s_0, \{s_3, s_4, s_5\})$$

δ':



mit der Zustandsüberführungstabelle:

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

Bestaunen Sie auch das **neue Animations-Feature** von XWizard am anderen Ende von diesem Link:

SKRIPT ID-C22298

Aufgabe 2	**12 Punkte**
2017-H-02	**Kellerautomaten**
	/ 12

Gegeben sei die folgende Sprache L (wobei $|w|_a$ wie immer die Anzahl der a's in w bezeichnet):

$$L = \{w \in \{a,b\}^\star \mid \exists n \in \mathbb{N}_0 : |w|_a = |w|_b + 2 \cdot n\}$$

L enthält also alle Wörter über $\{a,b\}$, mit folgenden Eigenschaften:

- es gibt **mindestens** so viele a's wie b's und
- falls es **mehr** a's als b's sind, dann ist dieser Überschuss $|w|_a - |w|_b$ eine gerade Zahl.

Bspw. gilt:

$$\lambda, aa, aaaa, ba, bbaaaa, baaaba, babaaaab \in L;$$
$$a, b, bbbaa, ababa, bbbbaa \notin L.$$

Geben Sie einen **nichtdeterministischen** Kellerautomaten $A = (E, S, K, \delta, s_0, k_0, F)$ an, mit $L(A) = L$. Geben Sie A vollständig an.

Hinweise:

- Zählen Sie beim Durchlaufen des Wortes zunächst den Überschuss $|w|_a - |w|_b$ im Keller mit und überprüfen Sie erst am Wortende, ob dieser gerade ist.
- Da unbekannt ist, wann das Wortende kommt, springen Sie einfach nichtdeterministisch an jeder (sinnvollen) Stelle „auf Verdacht“ in den Überprüfungsmodus.

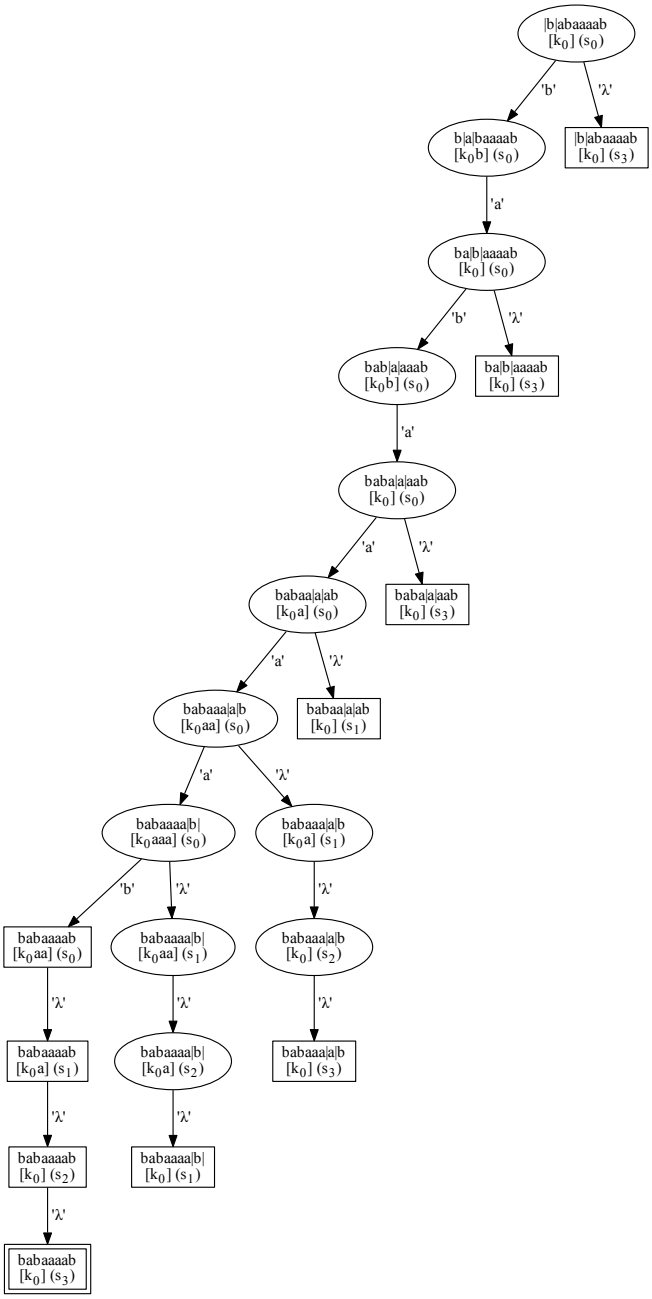
Lösung:

$$A = (\{a,b\}, \{s_0, s_1, s_2, s_3\}, \{a, b, k_0\}, \delta, s_0, k_0, \{s_3\})$$

$(s_0, a, k_0) \to \{(s_0, ak_0)\}$
$(s_0, b, k_0) \to \{(s_0, bk_0)\}$
$(s_0, a, a) \to \{(s_0, aa)\}$
$(s_0, b, b) \to \{(s_0, bb)\}$
$(s_0, a, b) \to \{(s_0, \lambda)\}$
$(s_0, b, a) \to \{(s_0, \lambda)\}$
$(s_0, \lambda, k_0) \to \{(s_3, k_0)\}$ // $|w|_a = |w|_b$ (falls Wort abgearbeitet)
$(s_0, \lambda, a) \to \{(s_1, \lambda)\}$ // Ab hier gilt: $|w|_a > |w|_b$
$(s_1, \lambda, a) \to \{(s_2, \lambda)\}$ // Zähle a's modulo 2
$(s_2, \lambda, a) \to \{(s_1, \lambda)\}$ // Zähle a's modulo 2
$(s_2, \lambda, k_0) \to \{(s_3, k_0)\}$ // Gerade Anzahl – alles gut.

SKRIPT ID-22238

Es ergibt sich für das Wort *babaaaab* folgender Berechnungsablauf:



Aufgabe 3 **9 Punkte**

2017-H-03 **Grammatiken**

/ 9

Gegeben sei eine Grammatik $G = (\{B, C, S, W, Z\}, \{a, b, c\}, P, S)$ mit

$$\begin{aligned} P = \{ & S \rightarrow aBC \mid aSBC, \\ & CB \rightarrow CZ, \\ & CZ \rightarrow WZ, \\ & WZ \rightarrow WC, \\ & WC \rightarrow BC, \\ & aB \rightarrow ab, \\ & bB \rightarrow bb, \\ & bC \rightarrow bc, \\ & cC \rightarrow cc\} \end{aligned}$$

SKRIPT ID-19536

Es gilt:

$$L(G) = \{a^n b^n c^n \mid n \in \mathbb{N}\} = \{abc, aabbcc, aaabbbccc, aaaabbbbcccc, \ldots\}$$

Gegeben sei außerdem die Sprache L_1, die ein b weniger pro Wort enthält als $L(G)$:

$$L_1 = \{a^n b^{n-1} c^n \mid n \in \mathbb{N}\} = \{ac, aabcc, aaabbccc, aaaabbbcccc, \ldots\}$$

(a) Kreuzen Sie an, von welchen Typen die Grammatik G ist:

rechtslinear	LR(k)	kontextfrei	kontextsensitiv	monoton	allgemein
			×	×	×

/ 2

(b) Leiten Sie das Wort *aabbcc* mit G ab.

/ 4

Lösung:

$$\overline{S} \Rightarrow a\overline{S}BC \Rightarrow aa\underline{B\overline{C}}BC \Rightarrow aaB\underline{\overline{C}\overline{Z}}C \Rightarrow aaB\underline{\overline{W}\overline{Z}}C \Rightarrow aaB\underline{\overline{W}\overline{C}}C \Rightarrow a\underline{a\overline{B}}BCC$$
$$\Rightarrow a\underline{a\overline{b}}BCC \Rightarrow aa\underline{b\overline{b}}CC \Rightarrow aabb\underline{c\overline{C}} \Rightarrow aabbcc$$

keine Abzüge geben (auch wenn der Stern fehlt).

(c) Ändern Sie G so ab, dass für die resultierende Grammatik G_1 gilt:

$$L(G_1) = L_1$$

Hinweise:

- Sie dürfen die Änderung(en) direkt durch Beschriftungen in G vornehmen oder verbal erklären, was geändert werden muss.
- Denken Sie nicht zu kompliziert! Es reicht eine kleine Anpassung mit „Korrektur“.

/ 3

Lösung:

SKRIPT ID-19537

$S \to aC$ und $S \to ac$ hinzufügen, $S \to aBC$ löschen.

Aufgabe 4	**8 Punkte**
2017-H-04	**Pumping Lemma für kontextfreie Sprachen**

/ 8

Gegeben sei die Sprache L_2:

$$L_2 = \{a^n b^{n-2} c^n \mid n \geq 2\} = \{aacc, aaabccc, aaaabbcccc, \ldots\}$$

Zeigen Sie mit dem Pumping-Lemma für kontextfreie Sprachen, dass L_2 nicht kontextfrei ist.

Hinweis: Ein passendes Pumpwort zu finden ist nicht schwer, aber ein „schlechtes“ Pumpwort kann den Beweis unmöglich machen.

Lösung: „Trickreiches“ Pumpwort $a^{n+2}b^n c^{n+2}$, Pumpen mit irgendeinem $i \neq 1$ führt dazu, dass sich die Anzahl höchstens zweier Zeichen ändert und mindestens eines Zeichens nicht usw. usf.

Aufgabe 5	**8 Punkte**
2017-H-05	**Berechenbarkeit**

/ 8

Das (nicht entscheidbare) **Halteproblem für Turingmaschinen H** kann als die Frage aufgefasst werden, ob eine gegebene Turingmaschine auf einer gegebenen Eingabe anhält.

(a) Formulieren Sie das Halteproblem für Turingmaschinen, wie man alle Entscheidungsprobleme in der Informatik definiert – als **formale Sprache L_H** (deren Entscheidbarkeit zu prüfen ist).

Hinweis: Sie dürfen umgangssprachlich schwammig (dabei aber bitte trotzdem **möglichst** präzise) formulieren. Denken Sie nicht zu kompliziert!

Lösung:

$L_H = \{\langle T, w\rangle \mid T$ hält auf w an$\}$: Die Menge aller Kodierungen $\langle T, w\rangle$ von Turingmaschinen T mit zugehörigen Eingaben w, sodass T auf w anhält.

/ 2

(b) Warum ist das Halteproblem für Turingmaschinen immerhin **semientscheidbar**?

Hinweis: Überlegen Sie, wie eine Turingmaschine U vorgehen kann, um bei Eingabe einer Turingmaschine T mit zugehöriger Eingabe w für T immer nach endlicher Zeit akzeptierend zu halten, wenn T auf w anhält (und im anderen Fall niemals akzeptierend zu halten).

/ 2

Lösung:

Eine universelle Turingmaschine U kann T auf w simulieren und genau dann in einen akzeptierenden Endzustand wechseln, wenn T anhält. Wir tun also eigentlich nichts anderes, als T zu „starten“ und abzuwarten, ob sie anhält. Tut sie das, sagen wir: Ok, Ausgabe ist *wahr*. Tut sie es aber nicht, wissen wir nichts weiter, denn sie könnte ja noch irgendwann anhalten. Wir können in diesem Fall also nur immer weiter warten – im nicht-haltenden Fall unendlich lang.

(c) Warum ist das **Halteproblem für endliche Automaten (Akzeptoren)** entscheidbar?

Lösung:

Weil sie, nachdem das letzte Zeichen von w eingelesen wurde, automatisch halten unabhängig davon ob w akzeptiert wird oder nicht.

/ 2

(d) **Zwei Zusatzpunkte:** Warum ist das **Halteproblem für linear beschränkte Automaten (LBA)** entscheidbar? ⟨Schwer! Aber wohlbekanntes Prinzip.⟩

Hinweis: Wie unterscheidet sich ein LBA von einer allgemeinen Turingmaschine – und wie kann dieser Unterschied ausgenutzt werden, um Endlosschleifen zu erkennen?

/ ⟨2⟩

Lösung:

Weil es durch die lineare Bandbeschränkung nur endlich viele mögliche Konfigurationen (also Kombinationen aus Zustand, Bandinschrift und Kopfposition) gibt. Tritt eine Konfiguration ein zweites Mal auf, muss die darauffolgende Rechnung genauso ablaufen wie zuvor, es ist also eine Endlosschleife eingetreten. Wir müssen nur die Turingmaschine für so viele Schritte simulieren, wie es verschiedene Konfigurationen gibt. Hat sie bis dahin angehalten – hat sie angehalten. Falls nicht, muss mindestens eine Konfiguration doppelt aufgetreten sein, also kann sie niemals anhalten.

(e) Das **Komplement des Halteproblems für Turingmaschinen $\overline{H}$** bezeichnet die Frage, ob eine gegebene Turingmaschine T auf einer gegebenen Eingabe w **nicht anhält**. Begründen Sie, dass, falls $\overline{H}$ entscheidbar wäre, H ebenfalls entscheidbar wäre.

Hinweis: Wie kann die Ausgabe einer Turingmaschine, die $\overline{H}$ entscheidet, ganz einfach in eine Ausgabe für H umgewandelt werden?

/ 2

Lösung:

Wäre $\overline{H}$ entscheidbar, dann gäbe es eine Funktion f (die charakteristische Funktion von $L_{\overline{H}}$) mit $f(\langle T, w\rangle) = 1$, falls T auf w nicht hält und $f(\langle T, w\rangle) = 0$, falls T auf w hält. Die Funktion g mit $g(x) = 1 - x$, wäre dann sehr leicht berechenbar und könnte verwendet werden um bei Eingabe $f(\langle T, w\rangle)$ die Aussage umzudrehen und somit das Halteproblem zu entscheiden. Offenbar ist also auch $\overline{H}$ **nicht entscheidbar**.

(f) **Zwei Zusatzpunkte:** Warum kann umgekehrt aus der Semientscheidbarkeit von H **nicht** auf die Semientscheidbarkeit von $\overline{H}$ geschlossen werden? ⟨Schwer!⟩

Hinweis: Denken Sie an Ihre Lösung zu (b). Warum funktioniert diese nicht für $\overline{H}$?

/ ⟨2⟩

Lösung:

Im Rahmen der Semientscheidbarkeit werden wahre und falsche Instanzen nicht gleich behandelt. Es reicht uns, für alle **wahren** Instanzen akzeptierend zu halten, um ein Problem „semientscheidbar“ zu nennen. Es reicht uns aber **nicht**, für alle **falschen** Instanzen nicht-akzeptierend zu halten. Das ist die wichtige Erkenntnis!

Daraus folgt, dass der Ansatz aus (b) für $\overline{H}$ nicht funktionieren kann, denn wir können nicht „abwarten“, ob T **nicht anhalten** wird – was hier den positiven Fall darstellt. Das Nicht-Anhalten klärt sich erst nach unendlich viel Zeit. Wir erkennen zwar nach endlicher Zeit die Fälle, in denen T irgendwann anhält, aber das reicht nicht aus, um Semientscheidbarkeit zu erreichen. Das sieht man auch daran, dass das Ergebnis einer Turingmaschine U, die – auf welche Art auch immer – H semientscheidet, nicht einfach umgedreht werden kann, um $\overline{H}$ semizuentscheiden. Denn der für $\overline{H}$ interessante, weil positive Fall, das T **nicht anhält**, würde unter Umständen „unendlich lang“ zum Berechnen benötigen, also besser gesagt überhaupt nicht berechnet werden.

Aufgabe 6 **6 Punkte**

2017-H-06 **Binary Decision Diagram**

/ 6

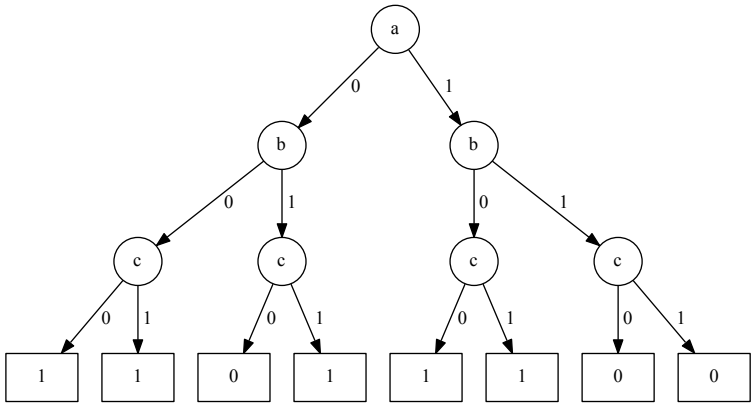
Gegeben sei die Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ durch folgende Wahrheitstabelle:

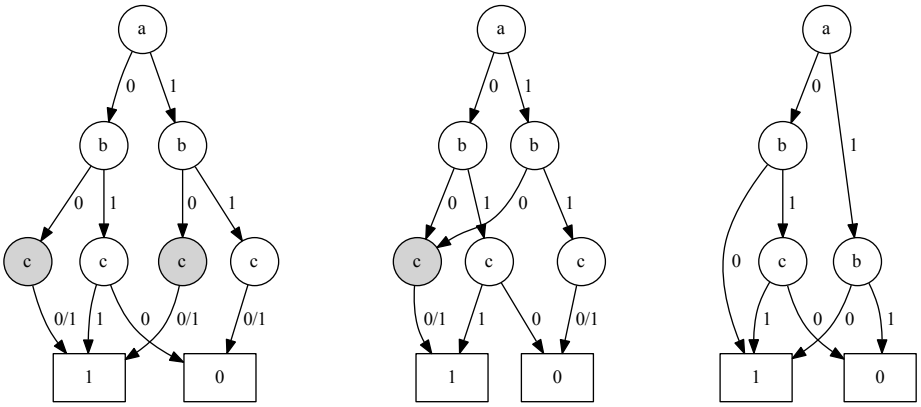
a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a) Erstellen Sie das zu f gehörende BDD bei Variablenreihenfolge $a \to b \to c$.

/ 5

Lösung:





SKRIPT ID-21377

(b) Geben Sie die Funktion f als Booleschen Term an.

/ 1

Lösung:

DNF aus dem BDD abgelesen:

$a'b' + a'bc + ab' == b' + a'bc$

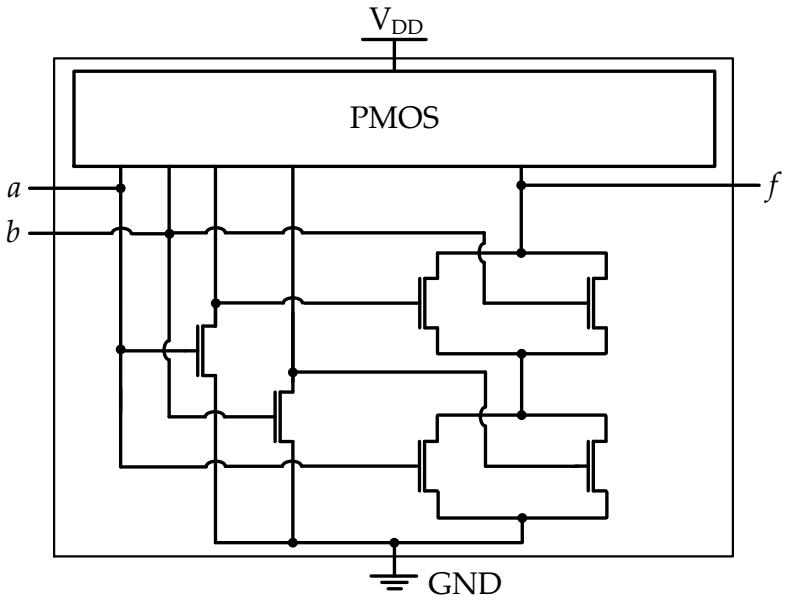
"Brute Force" aus der Wahrheitstabelle abgelesen:

$a'b'c' + a'b'c + a'bc + ab'c' + ab'c$

Aufgabe 7	**6 Punkte**
2017-H-07	**CMOS**

/ 6

Die Funktion $f : \mathbb{B}^2 \to \mathbb{B}$ sei durch folgende halb dargestellte CMOS-Schaltung gegeben:



(a) Lesen Sie aus dem gegebenen NMOS-Teil einen Booleschen Term für f ab.

/ 4

Lösung:

$$\begin{aligned} f(a,b) &= \neg((a \vee \neg b) \wedge (\neg a \vee b)) \\ &= \neg(a \vee \neg b) \vee \neg(\neg a \vee b) \\ &= (\neg a \wedge b) \vee (a \wedge \neg b) \end{aligned}$$

(b) Füllen Sie die gegebene Wahrheitstabelle aus und geben Sie an, um welche bekannte Boolesche Funktion es sich handelt.

Hinweis: Es reicht ein einziger bekannter Boolescher Operator.

/ 2

a	b	f
0	0	**Lösung:** 0
0	1	**Lösung:** 1
1	0	**Lösung:** 1
1	1	**Lösung:** 0

Lösung:

Es handelt sich um ein logisches XOR: $f = a \oplus b$.

Aufgabe 8	**10 Punkte**
2017-H-08	**Zahlendarstellung**

/ 10

Neben Dualzahlen sind auch **Oktalzahlen**, also Zahlen zur Basis 8, in der Informatik von großer Bedeutung. Gegeben sei daher eine Zahl Z in Gleitpunktdarstellung zur Basis 8, deren Dezimalwert Z_{10} in dieser Aufgabe bestimmt werden soll:

$$Z_{GPZ_8} = \underset{\text{Vorzeichen}}{1} \quad \underset{\text{Charakteristik}}{4\,2\,6} \quad \underset{\text{Mantisse}}{6\,2\,6\,7}$$

Hinweis: Zahlenwerte dieser Aufgabe müssen Sie nicht ausrechnen, es genügen korrekte Terme. Alle Einzelwerte müssen aber als Dezimalzahlen angegeben werden.

(a) Geben Sie zunächst für das **Dualsystem** die bekannte Formel zur Berechnung des Zahlenwerts $z(X_{GPZ_2})$ einer Gleitpunktzahl $X_{GPZ_2} = v\ c\ m$ sowie für das q der Exzess-q-Darstellung der Charakteristik an.

/ 2

Lösung:

$Z(x) = (-1)^v \cdot (1 + m) \cdot 2^{c-q}$ (Auch in Ordnung: $(-1)^v \cdot m \cdot 2^{c-q}$; um die implizite 1 geht es hier nicht.)

$q = 2^{n-k-1} - 1 = 2^{|c|-1} - 1$

(b) Welchen Wert hat nun das q für die Exzess-q-Darst. der Charakteristik von Z_{GPZ_8} (s. o.)?

Hinweis: Überlegen Sie, wie durch q der Zahlenbereich verschoben werden muss, um eine **möglichst gleiche Anzahl positiver und negativer darstellbarer Zahlen** in der Charakteristik zu erreichen. Orientieren Sie sich an der Formel für die Dualdarstellung.

/ 3

Lösung:

$q = 8^3/2 - 1 = 255$

Auf diese Weise wird die Skala der $8^3 = 512$ darstellbaren Zahlen:

$$\text{von } \underline{0} \text{ bis } 8^3 - 1 = \underline{511}$$

verschoben um etwa die Hälfte der darstellbaren Zahlen auf die weiterhin 512 Werte:

$$\text{von } 0 - q = \underline{-255} \text{ bis } 8^3 - 1 - q = \underline{256}$$

Bei einer naiven Umformung der binären Formel $q = 2^{L-1} - 1$ (wobei L die Länge der Charakteristik ist) könnte $q = 8^{3-1} - 1 = 63$ herauskommen. Dann ist man auf den Trick hereingefallen, der nur bei Binärzahlen funktioniert: nur hier bewirkt nämlich die Reduktion des Exponenten um 1 eine Halbierung des Ergebnisses.

(c) Welchen Wert hat der Exponent e von Z_{GPZ_8} (s. o.)? (Geben Sie den Wert in Abhängigkeit von q an, wenn Sie (b) nicht gelöst haben.)

/ 2

Lösung: $e = 4 \cdot 8^2 + 2 \cdot 8^1 + 6 \cdot 8^0 - q = 278 - 255 = 23$.

(d) Geben Sie den Dezimalwert von Z_{GPZ_8} (s. o.) an (in Abhängigkeit von q bzw. e, falls Sie diese Werte nicht berechnet haben).

Hinweis: In allen Darstellungen zur Basis ungleich 2 gibt es keine implizite führende 1.

/ 3

Lösung:

$$\begin{aligned} Z_{10} &= (-1)^1 \cdot 8^{23} \cdot (6 \cdot 8^0 + 2 \cdot 8^{-1} + 6 \cdot 8^{-2} + 7 \cdot 8^{-3}) \\ &= -8^{23} \cdot 6{,}357421875 \\ &\approx -3.752.759.497.495.286.906.880, \ldots \end{aligned}$$

SKRIPT ID-19518

Aufgabe 9	**8 Punkte**
2017-H-09	**Rechnerarchitektur**

/ 8

(a) Erläutern Sie, was man unter dem physikalischen von Neumann-Engpass versteht und beschreiben Sie kurz ein Konzept, wie dieser vermieden werden kann.

/ 3

Lösung:

Der physikalische von Neumann-Engpass bezeichnet das Problem, dass der Transport von Daten zwischen CPU und Arbeitsspeicher ein Vielfaches länger dauert als die Ausführung der Befehle durch die CPU und dass für jede Befehlsausführung zunächst mindestens ein Speicherzugriff erfolgen muss.

Möglichkeiten zur Vermeidung des von Neumann-Engpasses :

- Abbildung komplexer Datentypen (z.B. Matrix, Schlange, etc.) auf Hardware-Ebene, die direkt von der CPU verarbeitet werden können
- Verwendung eines Caches zur assoziativen Speicherung von Daten um die Zugriffszeiten zu verkürzen
- Parallelverarbeitung von Daten auf spezialisierter Hardware (z.B. Grafikkarten), wodurch mehrere Maschinenbefehle gleichzeitig ausgeführt werden können

(b) Erklären Sie kurz die Aufgaben der folgenden Protokolle

/ 3

- Transmission Control Protocol (TCP)

  Lösung:

  Stellt eine logische, verlässliche Verbindung zwischen Sender und Empfänger her.
- Internet Protocol (IP)

Lösung:

Sorgt für den unzuverlässigen, verbindungslosen Versand von Datenpakten durch das Internet.

- User Datagram Protocol (UDP)

  Lösung:

  Reicht den unzuverlässigen, verbindungslosen Dienst von IP an die Anwendungsschicht weiter.

(c) Geben Sie an, welches der beiden Protokolle TCP oder UDP Sie zum Streamen von Videodateien bevorzugen würden und begründen Sie Ihre Wahl kurz.

/ 2

Lösung:

UDP, da beim Streaming der Verlust einzelner Datenpakete nicht kritisch ist. UDP sendet auch bei Übertragungsfehlern einen beständigen Datenstrom, während sich bei TCP die Übertragungsrate dynamisch in Abhängigkeit auftretender Übertragungsfehler ändert.

Aufgabe 10 **8 Punkte**

2017-H-10 **Adressierungsarten**

/ 8

Die arithmetischen Befehle einer Assembler-Sprache seien folgendermaßen aufgebaut:

```
OpCode  Q1  Q2  Z
```

(Für Quelle 1, Quelle 2, Ziel.) Es gelten diese Kennzeichnungen für Adressierungsarten:

- Unmittelbare Adressierung: Präfix `#`
- Direkte Adressierung: ohne Präfix
- Indirekte Adressierung: Präfix `*`

Gegeben sei ein einfaches (niemals terminierendes) Assembler-Programm:

`I`	`loop`	`SUBTRACT`	`1`	`2`	`R3`	subtrahiert `Q1` minus `Q2`
`II`		`ADD`	`#1`	`#2`	`R4`	
`III`		`MULITPLY`	`*1`	`*2`	`R5`	
`IV`		`DIVIDE`	`9`	`*2`	`R10`	dividiert `Q1` durch `Q2`
`V`		`JUMP`	`loop`			springt bedingungslos nach `loop`

Gegeben sei außerdem ein aus zehn Registern bestehender Speicher mit folgenden ursprünglich gespeicherten Werten:

Reg.	Ursprungswert	Nach Zeile `I`	Nach Zeile `II`	Nach Zeile `III`	Nach Zeile `IV`
`R1`	5				
`R2`	4				
`R3`	6	**Lösung:** 1			
`R4`	2		**Lösung:** 3		
`R5`	3			**Lösung:** $9_{, 27, 81, \ldots}$	
`R6`	7				
`R7`	4				
`R8`	15				
`R9`	9				
`R10`	11				**Lösung:** 3

(a) Tragen Sie in Spalte „Nach Zeile *x*“ der Tabelle jeweils **den einen neuen Wert** ein, der sich bei Ausführung der Zeile *x* des Programms ergibt (Erstausführung der Schleife).

/ 4

(b) Wie ändert sich der Speicher im zweiten, dritten, vierten, … Schleifendurchlauf? Erklären Sie kurz den allgemeinen Ablauf.

/ 4

Lösung: Durch Zeile I und II ergeben sich dieselben Werte für R3 und R4 wie zuvor. In Zeile III entsteht allerdings statt R5·R4= $2 \cdot 3 = 9$ durch die neuen Werte in R4=3 und R5=9 nun der Wert $9 \cdot 3 = 27$ für R5. R10 bleibt unverändert. Da also alle Werte gleich bleiben, außer R5, wird R5 im nächsten Durchlauf beim erneuten Multiplizieren in Zeile III wieder um dem Faktor drei anwachsen, und so fort in jedem weiteren Durchlauf.

Aufgabe 11	**7 Punkte**
2017-H-11	**Betriebssysteme**

/ 7

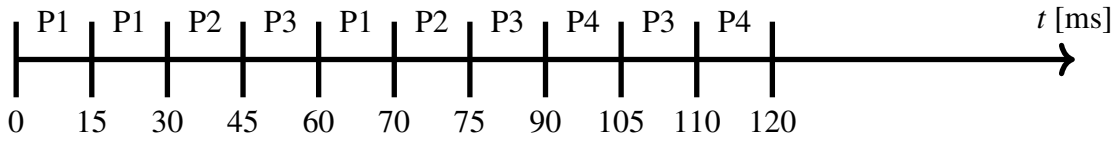
Betrachten Sie die Prozesse P1 bis P4, die **zeitverzögert** in die Warteschlange eines Prozessors zur Bearbeitung eingereiht werden. Die Ankunftszeit gibt an, zu welchem Zeitpunkt ein Prozess in die Warteschlange eingereiht wird.

Prozesse	CPU-Zeit in ms	Ankunftszeit
P1	40	0
P2	20	20
P3	35	40
P4	25	80

(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.

/ 3

Lösung:



(b) Für welche Zeitspannen $Z \geq 15$ ms wird P2 **vor allen anderen** Prozessen beendet? Geben Sie ein Intervall für alle möglichen Werte von Z an.

/ 2

Lösung:

$Z \in [20, 40)$

(c) Wie muss eine Folge an Prozessen, die nacheinander bei einem Prozessor ankommen, in Abhängigkeit der Bearbeitungszeiten angeordnet sein, damit die Zuteilungsverfahren **First Come First Serve** (FCFS) und **Shortest Job First** (SJF) diese in der gleichen Reihenfolge abarbeiten?

/ 2

Lösung:

Bezeichne $P = (P_1, \ldots, P_n)$ die Reihenfolge, in der die Prozesse am Prozessor ankommen und t_i die Berechnungsdauer für Prozess $P_i \in P$. FCFS und SJF liefern genau dann die gleichen Ergebnisse, wenn $t_i < t_j$ für alle $i < j$ und $i, j \in \{1, \ldots, m\}$ gilt. Die Prozesse müssen also aufsteigend sortiert entsprechend ihrer Bearbeitungszeit am Prozessor ankommen.