Karlsruher Institut für Technologie

aifb
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren

Lösung zur Klausur über den Stoff der Vorlesung **„Grundlagen der Informatik II“**

(90 Minuten)

Name: _____ Vorname: _____

Matr.-Nr.: _____ Semester: _____ (SS 2019)

Ich bestätige, dass ich die folgenden Angaben gelesen und mich von der Vollständigkeit dieses Klausurexemplars überzeugt habe (Seiten 1-18).

_____

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 **10 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	-	-	-	-	-	-	gesamt
(11)	(9)	(9)	(12)	(9)	(9)	(8)	(11)	(9)	(3)							(90)

Aufgabenübersicht

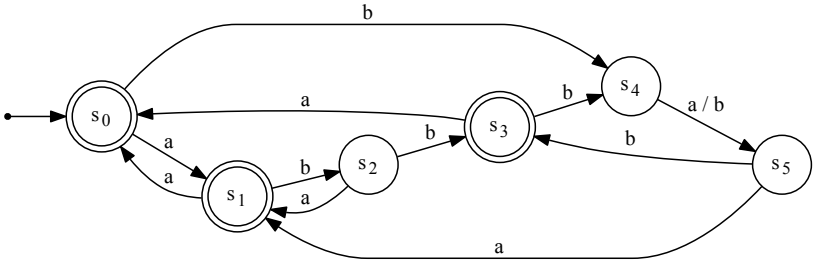
Aufgabe 1 **11 Punkte**

2019-N-01 **Minimierung endlicher Automaten**

/ 11

Gegeben sei folgender deterministischer endlicher Automat $A = (E, S, \delta, s_0, F)$. Durch das abgebildete Zustandsüberführungsdiagramm sei δ definiert.

δ :



(a) Minimieren Sie A und geben Sie den minimierten Automaten A' vollständig an. Geben Sie insbesondere die Übergangstabellen für A, A' und ein Zustandsüberführungsdiagramm δ' an.

/ 7

Lösung:

- Übergangstabelle für A:

	a	b
s_0	s_1	s_4
s_1	s_0	s_2
s_2	s_1	s_3
s_3	s_0	s_4
s_4	s_5	s_5
s_5	s_1	s_3

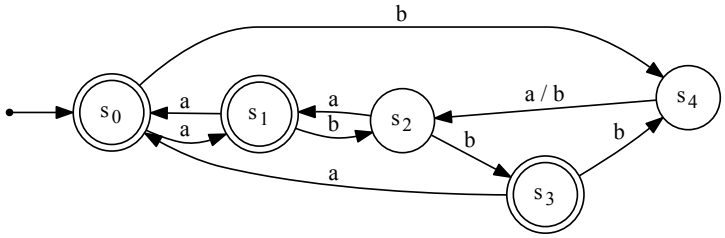
- Dreieckstabelle:

s_1	$\times_2$				
s_2	$\times_0$	$\times_0$			
s_3	$\times_3$	$\times_2$	$\times_0$		
s_4	$\times_0$	$\times_0$	$\times_1$	$\times_0$	
s_5	$\times_0$	$\times_0$	$-$	$\times_0$	$\times_1$
	s_0	s_1	s_2	s_3	s_4

- Übergangstabelle für A':

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

- Der minimierte Automat A' lautet wie folgt:
  $A' = (\{a, b\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta', s_0, \{s_0, s_1, s_3\})$



(b) Geben Sie die Mengen aller zueinander k-äquivalenten Zustände für $k \in \{0, 1, 2\}$ und die Mengen äquivalenter Zustände des endlichen Automaten A an. Verwenden Sie hierfür die nachfolgende Tabelle. Geben Sie auch einelementige Mengen an.

/ 4

Lösung:

0−Äquivalenz	$\{s_0, s_1, s_3\}, \{s_2, s_4, s_5\}$
1−Äquivalenz	$\{s_0, s_1, s_3\}, \{s_2, s_5\}, \{s_4\}$
2−Äquivalenz	$\{s_1\}, \{s_0, s_3\}, \{s_2, s_5\}, \{s_4\}$
Äquivalenz	$\{s_0\}, \{s_1\}, \{s_3\}, \{s_2, s_5\}, \{s_4\}$

Aufgabe 2 **9 Punkte**

2019-N-02 **Kellerautomat**

/ 9

Die Sprache $L \subseteq \{(,), [,]\}^*$ sei gegeben durch folgende Definition:

(a) $\lambda \in L$,

(b) falls $w, v \in L$, dann auch: wv, (w) und $[w] \in L$.

L ist also die Sprache aller wohlgeformten Klammerausdrücke über (,), [und].

Hinweis: Bspw. gilt:

$$\lambda, (((\,))), ()[](([][])), [](), ([]()) \in L;$$
$$)(, ()[], ()([]), ((,][, (]](][\notin L.$$

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

Lösung:

$A = (\{(,), [,]\}, \{s_0, s_1\}, \{k_0, (, [\}, \delta, s_0, k_0, \{s_0\})$

δ:
$\delta(s_0, (, k_0) \rightarrow \delta(s_1, (k_0)$
$\delta(s_0, [, k_0) \rightarrow \delta(s_1, [k_0)$
$\delta(s_1,), () \rightarrow \delta(s_1, \lambda)$
$\delta(s_1,], [) \rightarrow \delta(s_1, \lambda)$
$\delta(s_1, (, [) \rightarrow \delta(s_1, ([))$
$\delta(s_1, [, () \rightarrow \delta(s_1, [()$
$\delta(s_1, (, () \rightarrow \delta(s_1, (()$
$\delta(s_1, [, [) \rightarrow \delta(s_1, [[)$
$\delta(s_1, \lambda, k_0)) \rightarrow \delta(s_0, k_0)$

/ 7

(b) Zeigen Sie, dass Ihr Kellerautomat das Testwort ()[()] akzeptiert.

Lösung:

Erkennung des Testwortes ()[()]:

$(s_0, ()[()], k_0) \vdash (s_1,)[()], (k_0) \vdash (s_1, [()], k_0) \vdash (s_0, [()], k_0) \vdash (s_1, ()], [k_0) \vdash (s_1,)], ([k_0) \vdash (s_1,], [k_0) \vdash (s_1, \lambda, k_0) \vdash (s_0, \lambda, k_0)$

/ 2

Aufgabe 3	**9 Punkte**

2019-N-03 **Pumping Lemma**

/ 9

Gegeben sei die Sprache $L \subset 0^*$ durch $000 \in L$ und $w \in L \rightarrow w^{|w|} \in L$. Zeigen Sie mit Hilfe des Pumping Lemmas, dass L keine Typ 3 Sprache ist.

Hinweis: Es gilt zum Beispiel $000, 000000000 \in L$

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

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

(b) $y \neq \lambda$,

(c) $\forall i \in N : xy^iz \in L$.

Betrachte $i = 2$:
Es gilt offensichtlich $|w| < |xy^2z| \leq 2|xyz| = 2|w|$. Da das kleinste Wort in L drei Zeichen hat, gilt für beliebige Wörter w, $\tilde{w} \in L$, dass w und $\tilde{w}$ sich in der Länge mindestens um den Faktor drei unterscheiden. Also können zwei Wörter, deren Längen sich um höchstens den Faktor zwei unterscheiden (w und xy^2z), nicht beide gleichzeitig in L sein. Damit gilt $xy^2z < L$, wir sind zu einem Widerspruch gekommen und müssen die anfängliche Vermutung aufgeben, dass L vom Typ 3 ist.

Aufgabe 4 **12 Punkte**

2019-N-04 **Turingmaschine**

/ 12

Gegeben sei folgende Sprache:

$$L = \{w \in \{0,1\}^* \mid \exists n \in N_0 : w = (001)^n\}$$

Geben Sie eine Turingmaschine M an, die für Wörter $w \in L$ die gesamte Eingabe mit Einsen überschreibt sofern n gerade. Falls n ungerade soll die gesamte Eingabe mit Zweien überschrieben werden. Für alle anderen Wörter $w' \notin L$ soll die gesamte Eingabe mit Nullen überschrieben werden. Die Wörter bestehen trotzdem nur aus 0 und/oder 1. In allen Fällen soll M nach der Bearbeitung in einem Endzustand stoppen. Definieren Sie M vollständig.

Hinweis: Wenn die Eingabe das leere Wort ist, muss auch nichts überschrieben werden; die Maschine soll in diesem Fall nur in einem Endzustand halten.

Lösung:

$M = \{S, E, B, \delta, s_0, F\}$

$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}\}$

$E = \{0, 1\}$

$B = \{0, 1, 2, \#\}$

$F = \{s_0, s_{11}\}$

Zustand	0	1	#
s_0	(s_1,0,R)	(s_9,1,R)	
s_1	(s_2,0,R)	(s_9,1,R)	(s_9,#,L)
s_2	(s_9,0,R)	(s_3,1,R)	(s_9,#,L)
s_3	(s_4,0,R)	(s_9,1,R)	(s_8,#,L)
s_4	(s_5,0,R)	(s_9,1,R)	(s_9,#,L)
s_5	(s_9,0,R)	(s_6,1,R)	(s_9,#,L)
s_6	(s_1,0,R)	(s_9,1,R)	(s_7,#,L)
s_7	(s_7,1,L)	(s_7,1,L)	(s_{11},#,R)
s_8	(s_8,2,L)	(s_8,2,L)	(s_0,#,R)
s_9	(s_9,0,R)	(s_9,1,R)	(s_{10},#,L)
s_{10}	(s_{10},0,L)	(s_{10},0,L)	(s_{11},#,R)
s_{11}			

Aufgabe 5	**9 Punkte**
2019-N-05	**Komplexität**

/ 9

(a) Wie verhält sich die Menge der Sprachen, die von einer deterministischen Turing-Maschine in polynomieller Laufzeit erkannt werden können, zu der Menge der Sprachen, die von einer nichtdeterministischen Turing-Maschine in polynomieller Laufzeit erkannt werden können? Benennen Sie die Mengen.

/ 2

Lösung:

P ist die Komplexitätsklasse der Sprachen, die von deterministischen Turing-Maschinen erkannt werden können (0,5 Punkt)
NP ist die Komplexitätsklasse der Sprachen, die von nichtdeterministischen Turing-Maschinen erkannt werden können (0,5 Punkt)

P ist Teilmenge von NP: $P \subseteq NP$ (1 Punkt)

	w_1	w_2	w_3	w_4	w_5	...
M1	0	0	1	1	0	...
M2	1	0	1	0	0	...
M3	1	0	1	1	1	...
M4	0	1	1	1	0	...
⋮						

(b) Zeigen Sie durch Diagonalisierung, dass $p(E^*)$ für ein Alphabet E überabzählbar ist.

Hinweis: $p(E^*)$ entspricht der Menge aller Sprachen über den Wörtern aus E^*.

/ 7

Lösung:

Die Wörter aus E^* können abgezählt werden, bspw. durch eine längenlexikographische Sortierung:

$w_1 = \lambda$, w_2 = a, w_3 = b, w_4= aa, w_5= ab, w_6 = ba, w_7 = bb, ... für das Alphabet $\{a, b\}$. Wir können sie also als unendlich lange, aber abzählbare Liste anordnen.

Wir nehmen an, dass die Potenzmenge von E^* ebenfalls abgezählt werden kann. Dann kann jeder Teilnehmer $M \subseteq E^*$ (also $M \in SYMBOL(E^*)$) eindeutig einer natürlichen Zahl zugeordnet werden, d.h. man kann eine Liste abzählbar unendlicher Länge angeben, die an n-ter Stelle die n-te Teilmenge M_n enthält, wobei jedes Element von $SYMBOL(E^*)$ in der Liste enthalten ist.

Wir stellen folgende Tabelle T der Teilmengen von E^* auf (beispielhaft ausgefüllt):

In jeder Zelle $T_i j$ enthält die Tabelle eine Eins, wenn $w_j \in M_i$, und eine Null sonst. Eine Zeile i der Tabelle definiert also vollständig die Menge M_i. Betrachte die Menge

$M' = \{w_j | w_j \notin Mj\} \subseteq E*$ mit $j \in \mathbb{N}$

In obigem Beispiel enthielt M' w_1 und w_2 (und u.U. weitere Elemente w_j mit $j > 4$), aber nicht w_3 und w_4. Unserer Annahme entsprechend müsste für ein $n \in \mathbb{N}$ gelten $M' = M_n$, da alle Teilmengen von E^* in der Liste vorkommen sollten. Die Menge M' ist aber gerade so definiert, dass sie sich in mindestens einem Element (dem Diagonalelement) von jeder Menge in der Liste unterscheidet.

Da es (mind.) eine Menge gibt, die in der Liste nicht enthalten ist, muss unsere Anfangsannahme falsch sein, und die Menge $SYMBOL(E^*)$ ist nicht abzählbar, sondern überabzählbar.

Aufgabe 6 **9 Punkte**

2019-N-06 **Mealy-/Moore-Automat**

/ 9

Im Folgenden soll ein Automat modelliert werden, der Golfbälle zum Trainieren ausgibt. Der zu modellierende Automat nimmt ausschließlich 1€ und 2€ Münzen an. Maximal wird vom Automaten eine Gesamteingabe von 3€ akzeptiert. Sofern dieser Betrag überschritten wird, wird die Differenz direkt zurückgegeben. Über eine Rückgabetaste kann der Kunde sich das eingezahlte Geld direkt zurückgeben lassen und den Vorgang abbrechen. Kunden können zwischen drei verschiedenen Optionen wählen. Für jede dieser Optionen gibt es am Automaten eine Taste:
1) Die Ausgabe von 10 Golfbällen kostet 1€.
2) Die Ausgabe von 25 Golfbällen kostet 2€.
3) Die Ausgabe von 50 Golfbällen kostet 3€.

Konstruieren Sie den im folgenden Abschnitt beschriebenen Ballautomaten als Mealy-Automat M. Geben Sie M vollständig an. Benutzen Sie dafür das folgende Ein- und Ausgabealphabet (auf der nächsten Seite).

Hinweis: Bei einer Eingabe von beispielsweise 3€ kann der Kunde sich direkt 50 Golfbälle ausgeben lassen. Es ist jedoch auch möglich, sich erst 25 Golfbälle ausgeben und den Restbetrag von 1€ auszahlen zu lassen oder mit diesem Restbetrag 10 weitere Golfbälle zu ordern.

Eingabealphabet $E = \{M_1, M_2, M_U, 10, 25, 50, M_R\}$ mit der Bedeutung:

M_1	1 Euro Münze
M_2	2 Euro Münze
M_u	Ungültige Münze
10	Taste zur Ausgabe von 10 Golfbällen
25	Taste zur Ausgabe von 25 Golfbällen
50	Taste zur Ausgabe von 50 Golfbällen
M_R	Rückgabetaste

Ausgabealphabet $A = \{RU, R1, R2, R3, A10, A25, A50, -\}$ mit der Bedeutung:

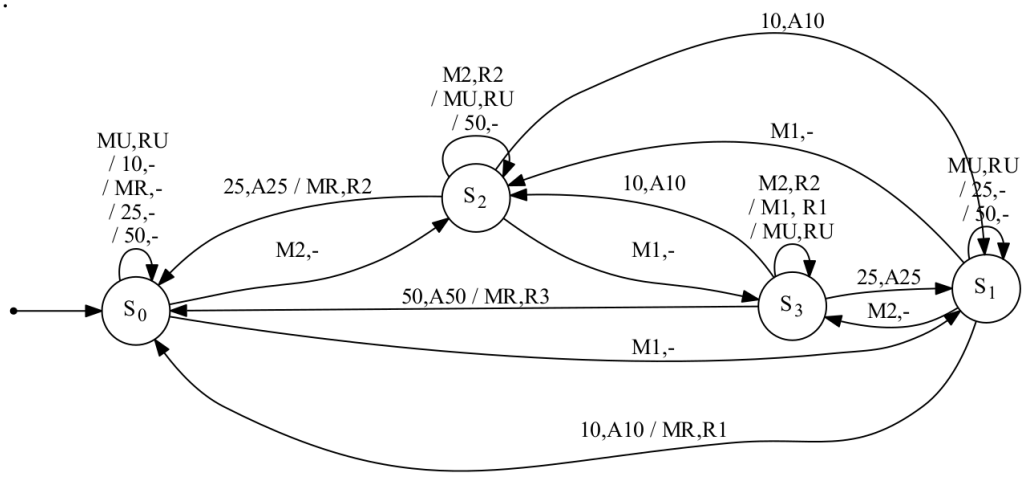
RU	Rückgabe von ungültiger Münze
R1	Rückgabe von Einzahlungen in Höhe von 1 Euro
R2	Rückgabe von Einzahlungen in Höhe von 2 Euro
R3	Rückgabe von Einzahlungen in Höhe von 3 Euro
A10	Ausgabe von 10 Golfbällen
A25	Ausgabe von 25 Golfbällen
A50	Ausgabe von 50 Golfbällen
-	Keine Aktion

Lösung:

Zustandsmenge $S = \{s0, s1, s2, s3\}$ mit der Bedeutung:
s_0 = Kein Geld eingeworfen
s_1 = Einzahlungen in Höhe von 1 Euro
s_2 = Einzahlungen in Höhe von 2 Euro
s_3 = Einzahlungen in Höhe von 3 Euro

Folglich ergibt sich folgender Mealy-Automat:
$M = (\{M_1, M_2, M_U, 10, 25, 50, M_R\}, s_0, s_1, s_2, s_3, RU, R1, R2, R3, A10, A25, A50, -, \delta, \gamma, s_0)$

δ, γ:



Aufgabe 7	**8 Punkte**
2019-N-07	**Huffman**

/ 8

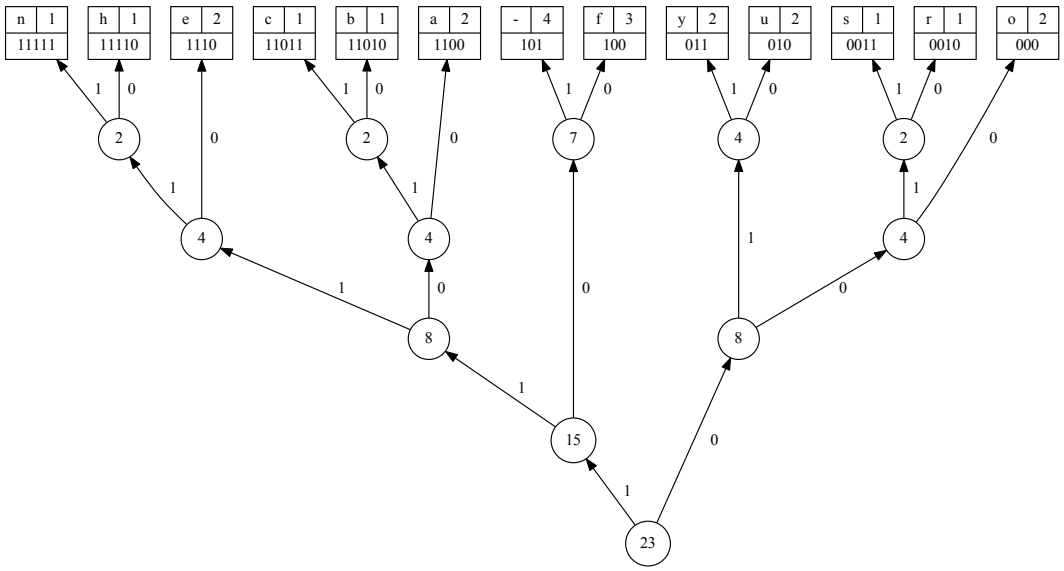
Gegeben sei der folgende Text aus 23 Zeichen:

before-you-can-say-huff

Erzeugen Sie zu der durch den Text gegebenen Wahrscheinlichkeitsverteilung eine Huffman-Kodierung. Tragen Sie dazu die Häufigkeiten der Zeichen in die untere Zeile der ersten Tabelle ein, erstellen Sie einen Huffman-Baum mit Angabe der Häufigkeiten an den Knoten und geben Sie in der zweiten Tabelle für jedes Zeichen eine dem Baum entsprechende Kodierung an.

n	h	e	c	b	a	-	f	y	u	s	r	o
1	1	2	1	1	2	4	3	2	2	1	1	2

Lösung (Beispiel):



Aufgabe 8 **11 Punkte**

2019-N-08 **Binary Decision Diagram**

/ 11

Gegeben sei die Boolesche Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ mit

$$f(a, b, c) = \overline{(a \oplus b)} \wedge c$$

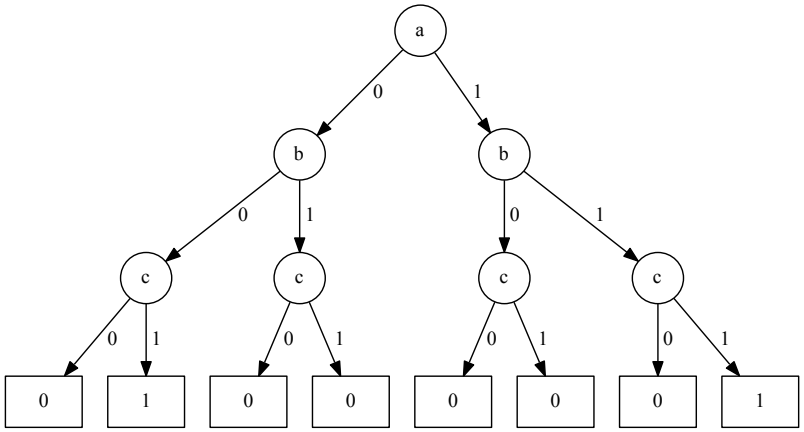
(a) Tragen Sie in die folgende Tabelle die Funktionswerte der Funktion f ein.

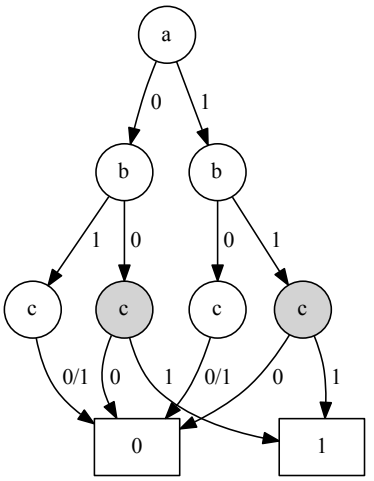
/ 3

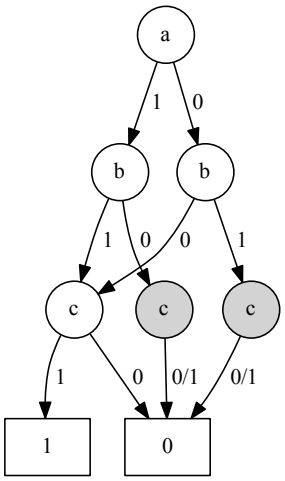
a	b	c	$a \oplus b$	$\overline{(a \oplus b)}$	f
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	0	1	0
1	1	1	0	1	1

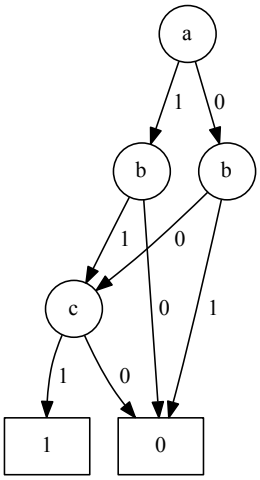
(b) Berechnen Sie das zu f gehörende Binary Decision Diagram (BDD) mit der Variablenreihenfolge $a \to b \to c$, indem Sie mit einer Baumdarstellung der Wertetabelle beginnen und den Algorithmus aus der Vorlesung anwenden.

/ 7




a
0
1
b
b
1
0
0
1
c
c
c
c
0/1
0
1
0/1
0
1
0
1


a
1
0
b
b
1
0
0
1
c
c
c
1
0
0/1
0/1
1
0


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

(c) Geben Sie die disjunktive Normalform (DNF): f (a, b, c) an.
f(a, b, c)= abc + a'b'c

/ 1

Aufgabe 9 **9 Punkte**

2019-N-09 **Programmiersprachen**

/ 9

In der Vorlesung haben wir gesehen, dass alle gängigen Programmiersprachen dieselbe Ausdrucksmächtigkeit haben, also einander simulieren können. Im Folgenden seien gegeben:

- eine **GOTO-Sprache** (das sei eine übliche Assemblersprache, die Sprünge in Abhängigkeit des aktuellen Akkumulator-Werts erlaubt: `JNZ label`) und
- eine **WHILE-Sprache** (das sei eine höhere Sprache vergleichbar zu Java, die allgemeine WHILE-Schleifen unterstützt: `while (b) { * do something * }`).

(a) Gegeben sei ein Programmausschnitt in der GOTO-Sprache:

/ 4

```
label:  LOAD R1
        ADD R2
        STORE R1
        LOAD R4
        ADD R1
        STORE R4
        LOAD R3
        SUB #1
        STORE R3
        JNZ label  // „JUMP NOT ZERO“
```

Simulieren Sie das Programm durch die WHILE-Sprache, d. h. geben Sie ein äquivalentes WHILE-Programm an.

Hinweise:

- Nutzen Sie möglichst Java-Notation oder einen verständlichen Pseudocode.
- Der Akkumulator sei über `ACC` erreichbar, die Register über ihre Namen `Ri`.

Lösung:

```
while (R3 != 0) {
    R1 += R2;
    R4 += R1;
    R3 -= 1;
}
```

(b) Was müsste im gegebenen Programmausschnitt der **GOTO-Sprache** verändert werden, damit bei Terminierung die Summe $\sum_{x=0}^{4} 2^x$ in `R4` steht?

/ 5

Hinweis: $\sum_{x=0}^{4} 2^x = 2^0 + 2^1 + 2^2 + 2^3 + 2^4$. Sie können die Lösung auch im gegebenen Code eintragen.

Lösung:

```
         LOAD #1
         STORE R1
         LOAD #2
         STORE R2
         LOAD #4
         STORE R3
         LOAD #1
         STORE R4
label:   LOAD R1
         MULT R2
         STORE R1
         LOAD R4
         ADD R1
         STORE R4
         LOAD R3
         SUB #1
         STORE R3
         JNZ label  // „JUMP NOT ZERO“
```

Aufgabe 10 **3 Punkte**

2019-N-10 **Betriebssysteme**

/ 3

Betrachten Sie die Prozesse P1 bis P4, die nacheinander in die Warteschlange eines Prozessors zur Bearbeitung eingereiht werden.

Prozesse	CPU-Zeit in ms
P1	20
P2	10
P3	25
P4	35

Teilen Sie den Prozessen Rechenzeit gemäß dem Round-Robin Verfahren zu. Die Zeitscheibe sei dabei in feste Zeitspannen der Länge $Z = 15\,\text{ms}$ unterteilt. Veranschaulichen Sie Ihr Ergebnis auf dem gegebenen Zeitstrahl.

Lösung: