Karlsruher Institut für Technologie

Institut für Angewandte Informatik
und Formale Beschreibungsverfahren

17.02.2014

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

Name: _____ Vorname: _____

Matr.-Nr.: _____ Semester: _____ (WS 2013/14)

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)	(7)	(8)	(11)	(10)	(10)	(8)	(9)	(7)	(5)	(6)						(90)

Aufgabenübersicht

Aufgabe 1 **9 Punkte**

2014-H-01 **Endliche Automaten**

/ 9

Mit Hilfe endlicher Automaten sollen Bitstrings $B \in \{0,1\}^\star$ darauf geprüft werden, ob es sich um Zahlen im BCD-Format handelt. Die Bitstrings werden beginnend mit der höchstwertigen Stelle seriell abgearbeitet. Geben Sie die Automaten in den folgenden Teilaufgaben jeweils vollständig an.

Hinweis: Für jeden 4-Bit-Block $B = b_3b_2b_1b_0$, der eine einzelne Ziffer Z kodiert, gilt: Beginnt B mit einer 0, handelt es sich auf jeden Fall um gültigen BCD-Code. Falls b_3 eine 1 ist, müssen b_2 und b_1 gleich 0 sein.

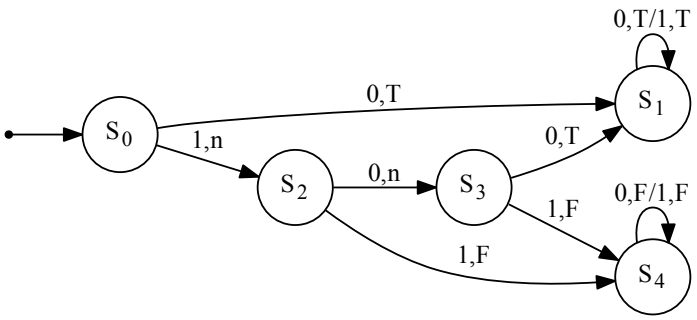
(a) Geben Sie zunächst einen Mealy-Automaten $EA_1 = (\{0,1\}, S, \{-, T, F\}, \delta, \gamma, s_0)$ an, der ausgibt, ob es sich bei einer vierstelligen Bitfolge $B \in \{0,1\}^4$ um eine einzelne BCD-kodierte Ziffer Z handelt oder nicht. Verwenden Sie dabei das Ausgabealphabet wie folgt:

- $-$: noch keine Aussage möglich,
- T : *true*, korrekte BCD-Ziffer (sobald erkannt),
- F : $false$, keine BCD-Ziffer (sobald erkannt).

Hinweis: Beachten Sie, dass für jedes der vier gelesenen Bits die Ausgabe dem jeweils aktuellen Stand der Überprüfung entsprechen soll. Eingaben mit mehr als vier Zeichen müssen hier noch nicht betrachtet werden.

Lösung:

$$EA_1 = (\{0,1\}, \{s_0, s_1, s_2, s_3, s_4\}, \{T, F, -\}, \delta, \gamma, s_0)$$

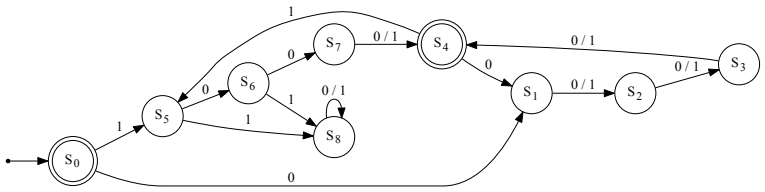


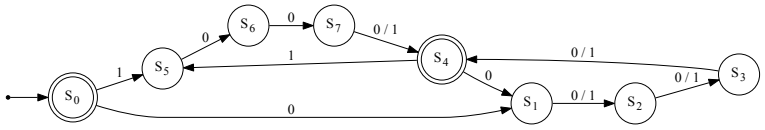
/ 4

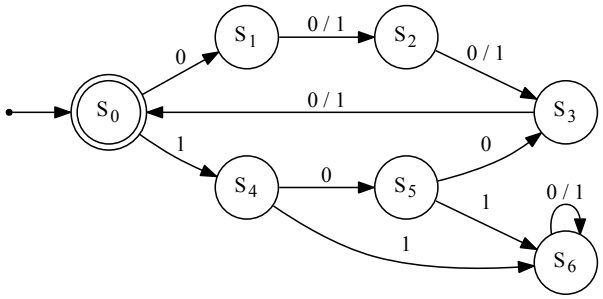
(b) Geben Sie nun einen Akzeptor $EA_2 = (\{0,1\}, S, \delta, s_0, F)$ an, der nicht nur einzelne Ziffern, sondern beliebig lange Bitfolgen $B \in \{0,1\}^\star$ einliest und immer dann einen Endzustand erreicht, wenn die bisher eingelesene Bitfolge einer (möglicherweise mehrstelligen) Zahl in gültiger BCD-Kodierung entspricht.

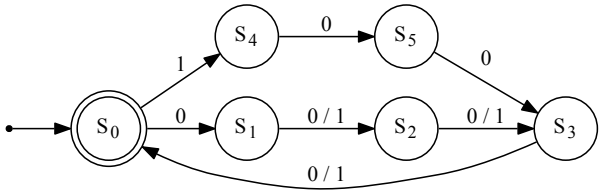
Lösung:

$$EA_2 = (\{0,1\}, \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8\}, \delta, s_0, \{s_0, s_4\})$$









/ 5

Aufgabe 2	**7 Punkte**
2014-H-02	**Pumping-Lemma**

/ 7

(a) Welche Idee liegt den Pumping-Lemmata zugrunde und wie unterscheidet sich hierbei das Pumping-Lemma für EA-Sprachen von dem für kontextfreie Sprachen? Erklären Sie in eigenen Worten und zeichnen Sie insbesondere zu beiden jeweils eine typische Skizze, um die Grundidee zu verdeutlichen.

/ 4

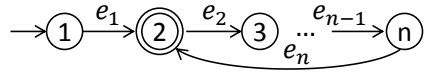
Lösung:

Die grundlegende Idee des Pumping-Lemmas ist, dass bei k unterscheidbaren Ereignissen (bspw. Zuständen eines Automaten) in jeder Folge von $n > k$ Ereignissen mindestens ein Ereignis doppelt vorkommen muss. Wenn der Übergang von einem Ereignis zum nächsten jeweils nur von dem aktuellen Ereignis abhängt (was wegen des Fehlens zusätzlicher Speicher bei endlichen Automaten der Fall ist, im erweiterten Sinn aber auch bei kontextfreien Grammatiken, s.u.), ergibt sich daraus die Möglichkeit, die Ereignisfolge zwischen dem doppelten Auftreten eines Ereignisses beliebig oft zu wiederholen.

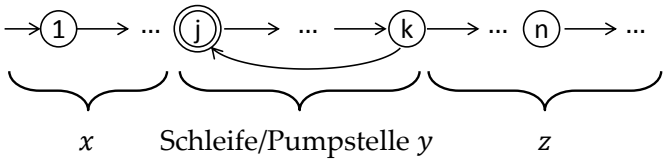
Bei dem Pumping-Lemma für EA-Sprachen gilt: Um Wörter w zu akzeptieren, die aus mindestens so vielen Zeichen bestehen wie es Zustände gibt, benötigt der Automat mindestens eine Schleife.

Bei dem für kontextfreie Sprachen bezieht sich die Ereignisfolge auf die Nichtterminalzeichen auf einem Pfad des Ableitungsbaums für ein ausreichend langes Wort der Sprache. Das heißt, für ein ausreichend langes Wort der Sprache enthält der Ableitungsbaum mindestens einen Pfad, der länger ist als die Anzahl der Nichtterminalzeichen der Grammatik. Es gibt mindestens einen Weg von der Wurzel zu einem Blatt, auf dem eine Variable mehrfach auftritt. Der Teil zwischen dem ersten und zweiten Auftreten von A kann beliebig oft wiederholt werden.

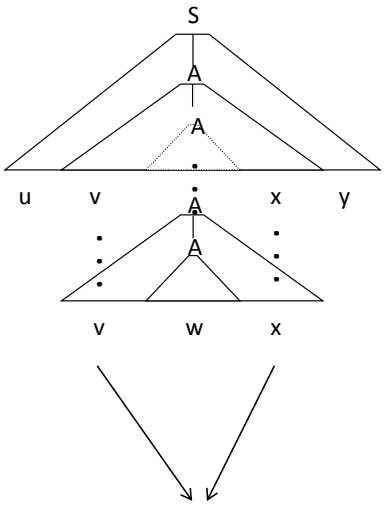
Bei n Zuständen eines EAs muss sich beim Durchlaufen der Zustände nach **spätestens** n Schritten ein Zustand wiederholen:



Allgemein gilt:



Bei Typ-2-Sprachen gilt dieselbe Argumentation auf Ebene der Nonterminalsymbole einer kontextfreien Grammatik.



Zwei Pumpstellen

(b) Entscheiden Sie, welche der folgenden Aussagen richtig sind und welche falsch. Für jede richtige Antwort erhalten Sie 0,5 Punkte, für jede falsche 0,5 Punkte Abzug. In Summe erhalten Sie keine negative Punktzahl für diese Teilaufgabe.

/ 3

Lösung:

	Wahr	Falsch
Die Aussage des Pumping Lemmas für EA-Sprachen (bezüglich der Existenz einer Pumpstelle in ausreichend langen Wörtern einer Sprache) ist nur für EA-Sprachen erfüllt.	☐	☒
Für eine Sprache gilt entweder nur das Pumping-Lemma für EA-Sprachen oder nur das für kontextfreie.	☐	☒
Die Existenz von zwei Pumpstellen gemäß Pumping-Lemma für kontextfreie Sprachen ist sowohl notwendig als auch hinreichend, um eine Sprache als kontextfrei zu charakterisieren.	☐	☒
Wenn für eine Sprache die Aussage des Pumping-Lemmas für kontextfreie Sprachen erfüllt ist, dann gilt für diese Sprache auch die Aussage des Pumping-Lemmas für EA-Sprachen.	☐	☒
Die Aussage des Pumping-Lemmas für kontextfreie Sprachen gilt auch für jede reguläre Sprache.	☒	☐

Die Aussage beider Pumping-Lemmata ist, dass es eine Zahl $n \in \mathbb{N}$ gibt, sodass für jedes Wort w mit $\|w\| \leq n$ eine Zerlegung $z = uvw$ existiert, mit $\|uv\| \leq n$, $\|v\| \geq 1$ und $uv^i w \in L$ für alle $i \in \mathbb{N}_0$.	☐	☒

Aufgabe 3 **8 Punkte**

2014-H-03 **Grammatiken**

/ 8

Gegeben sei die Grammatik

$$G = (\{S, S', A, B, X, Y, Z\}, \{a\}, P, S),$$

wobei P durch die erste Spalte folgender Tabelle gegeben sei:

Produktion	rechtslinear	kontextfrei	kontextsensitiv	monoton
$S \to \lambda$	×	×		
$S \to a$	×	×	×	×
$S \to S'AAX$		×	×	×
$S' \to S'A$		×	×	×
$S' \to ZB$		×	×	×
$BA \to YAB$				×
$YA \to AY$				×
$BX \to \lambda$				
$BX \to BAX$			×	×
$A \to a$	×	×	×	×
$Y \to a$	×	×	×	×
$Z \to \lambda$	×	×		

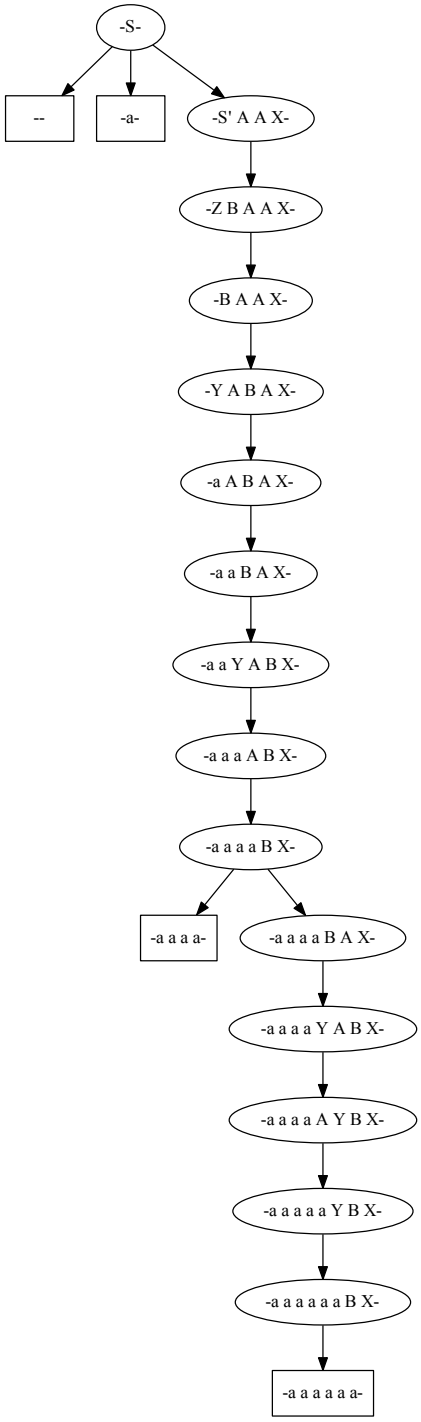
(a) Kreuzen Sie für jede Regel in der Tabelle an, ob sie rechtslinear, kontextfrei, kontextsensitiv und/oder monoton ist.

/ 4

$$
\begin{aligned}
G &= (\{S, S', A, B, X, Y, Z\}, \{a\}, P, S) \\
P &= \{S \to \lambda \mid a \mid S'AAX, \\
&\quad S' \to S'A \mid ZB, \\
&\quad BA \to YAB, \\
&\quad YA \to AY, \\
&\quad BX \to BAX \mid \lambda, \\
&\quad A \to a, \\
&\quad Y \to a, \\
&\quad Z \to \lambda\}
\end{aligned}
$$

(b) Leiten Sie vier Wörter $w_1, \ldots, w_4 \in L(G)$ mit der Grammatik G ab.

/ 4


-S-
--
-a-
-S' A A X-
-Z B A A X-
-B A A X-
-Y A B A X-
-a A B A X-
-a a B A X-
-a a Y A B X-
-a a a A B X-
-a a a a B X-
-a a a a-
-a a a a B A X-
-a a a a Y A B X-
-a a a a A Y B X-
-a a a a a Y B X-
-a a a a a a B X-
-a a a a a a-

Aufgabe 4 **11 Punkte**

2014-H-04 **Kellerautomaten**

/ 11

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

$$L = \left\{ w \in E^\star \mid w = 0^n u \text{ mit } n \in \mathbb{N}_0 \text{ und } u \in \{0,1\}^n \cup \{0,1\}^{2n} \right\}$$

Wörter aus L können also unterteilt werden in einen Anfangsteil aus n Nullen, auf den ein Endteil u aus Nullen und Einsen folgt, der entweder $|u| = n$ oder $|u| = 2n$ lang ist. Es gilt beispielsweise (Leerstellen kennzeichnen bei Wörtern aus L beispielhaft eine mögliche Unterteilung in Anfangs-Nullen und u und gehören nicht zur Sprache):

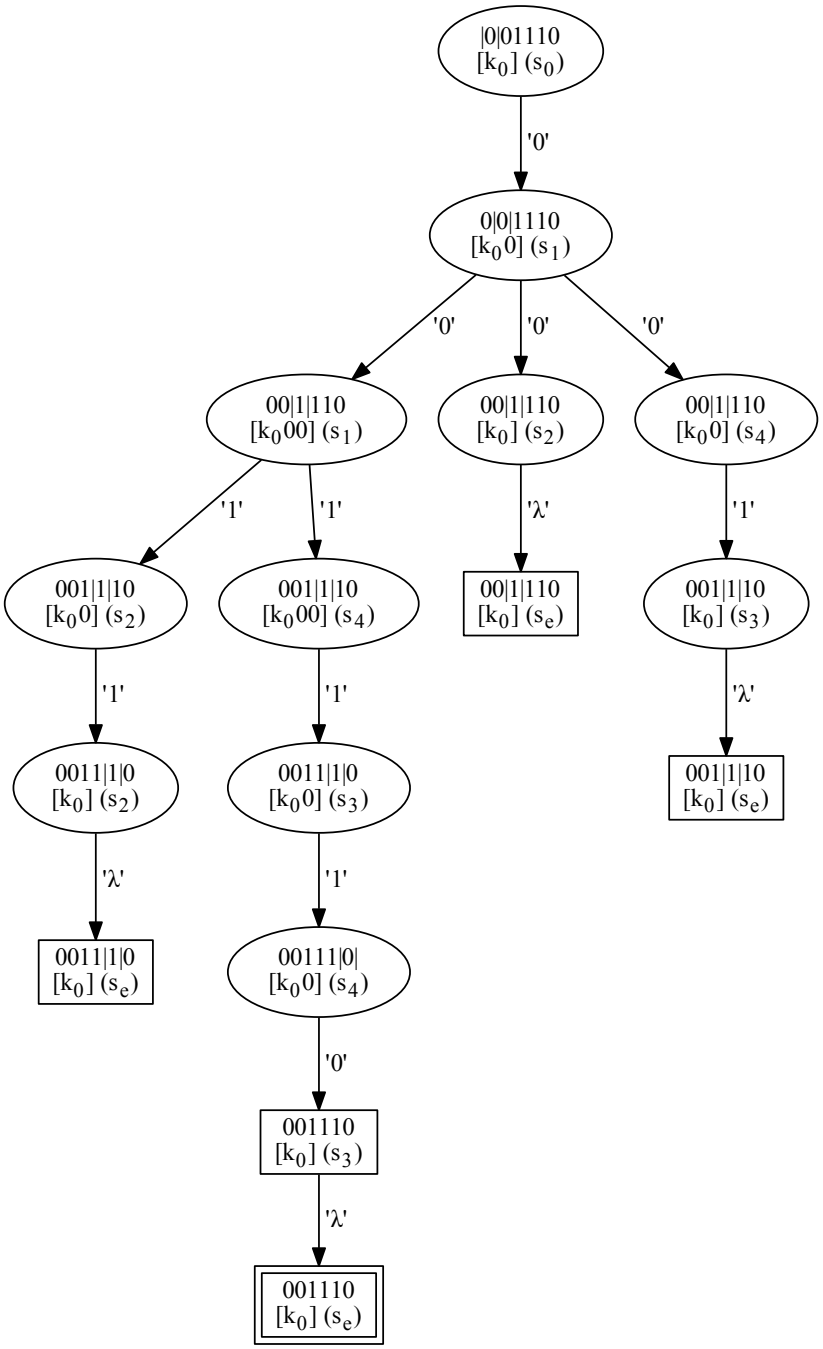
$$\lambda, 0\ 11, 00\ 00, 00\ 11, 00\ 0010, 000\ 111 \in L;$$
$$100, 00000, 100000, 0101 \notin L$$

(a) 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.

/ 9

Lösung: $A = (\{0,1\}, \{s_0.s_1, s_2, s_3, s_4, s_e\}, \{k_0, 0\}, \delta, s_0, k_0, \{s_0, s_e\})$

$$\delta(s_0, 0, k_0) = \{(s_1, 0k_0)\}$$
$$\delta(s_1, 0, 0) = \{(s_1, 00), (s_2, \lambda), (s_4, 0)\}$$
$$\delta(s_1, 1, 0) = \{(s_2, \lambda), (s_4, 0)\}$$
$$\delta(s_2, 0, 0) = \{(s_2, \lambda)\}$$
$$\delta(s_2, 1, 0) = \{(s_2, \lambda)\}$$
$$\delta(s_2, \lambda, k_0) = \{(s_e, k_0)\}$$
$$\delta(s_4, 0, 0) = \{(s_3, \lambda)\}$$
$$\delta(s_4, 1, 0) = \{(s_3, \lambda)\}$$
$$\delta(s_3, 0, 0) = \{(s_4, 0)\}$$
$$\delta(s_3, 1, 0) = \{(s_4, 0)\}$$
$$\delta(s_3, \lambda, k_0) = \{(s_e, k_0)\}$$



(b) Wäre es auch möglich, für die Sprache L einen deterministischen Kellerautomaten anzugeben? Begründen Sie in kurzen Worten.

/ 2

Lösung:

Nein, hier wird ein nichtdeterministischer Kellerautomat gebraucht. Da der Kellerautomat zuerst gewährleisten muss, dass nur Nullen gelesen werden, und aber mit jeder Null auch die „Umkehrstelle“ erreicht sein kann, muss der Automat nichtdeterministisch sein. Zudem muss der Automat im zweiten Wortteil auch noch überprüfen, ob es sich um n oder $2n$ Zeichen handelt.

Aufgabe 5 **10 Punkte**

2014-H-05 **Komplexitätstheorie**

/ 10

Gegeben seien die Probleme SAT (das Erfüllbarkeitsproblem aussagenlogischer Formeln) und $CLIQUE$ (die Frage, ob ein Graph einen vollständig verbundenen Teilgraphen mit k Knoten enthält). Gegeben sei weiter die Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ mit

$$f(a, b, c) = \neg abc + a\neg bc + ab\neg c + \neg a\neg b\neg c$$

(a) Erklären Sie am Beispiel der Funktion f, wie man die Erfüllbarkeit einer aussagenlogischen Formel prüfen kann. Ist f erfüllbar?

/ 2

Lösung: Beispielsweise eine Wertetabelle aufstellen und nach einer 1 suchen oder nacheinander alle möglichen Wertekombinationen für die Variablen a, b, c eingeben und schauen, ob eine 1 herauskommt usw. f ist erfüllbar, da $f(0, 0, 0) = 1$.

(b) Warum ist für die angegebene Darstellung von f die Erfüllbarkeit verhältnismäßig leicht (in Linearzeit bzgl. Variablenanzahl) zu überprüfen?

/ 1

Lösung: Weil die Formel in DNF angegeben ist, was bedeutet, dass man die Klauseln einzeln prüfen kann und nur Linearzeit benötigt wird.

(c) Angenommen, Sie wissen, dass SAT NP-vollständig ist. Wie können Sie argumentieren, dass $CLIQUE$ ein schwieriges Problem ist?

- Wie kann man unter Nutzung des Wissens, dass SAT NP-vollständig ist, zeigen, dass $CLIQUE$ NP-schwer ist? Konkretisieren Sie die allgemeine Definition der Polynomialzeitreduktion auf die Ebene aussagenlogischer Formeln und Graphen.

  Lösung: Man muss SAT auf $CLIQUE$ reduzieren, also eine polynomialzeitberechenbare Funktion finden, die aus einer aussagenlogischen Formel einen Graphen und eine zugehörige Zahl k berechnet, sodass genau dann eine Clique der Größe k im Graphen vorhanden ist, wenn die Formel erfüllbar ist (eine Rückführung der Lösung ist dann nicht notwendig).

- Was folgt, wenn Sie den Beweis geführt haben? Welche Beziehung besteht zwischen den Schwierigkeitsgraden von SAT und $CLIQUE$?

  Lösung: Aus dem Beweis folgt, dass $CLIQUE$ mindestens so schwer ist wie SAT (bis auf polynomielle Faktoren). Wenn SAT also mindestens Exponentialzeit zur Lösung benötigt (wie heute angenommen wird), dann gilt das auch für $CLIQUE$. Falls $P = NP$, wären beide Probleme aber deutlich leichter als angenommen.

/ 5

(d) Angenommen, Sie kennen keine NP-schweren Probleme. Welche Schritte müssten Sie ausführen, um zu zeigen, dass SAT NP-schwer ist?

/ 2

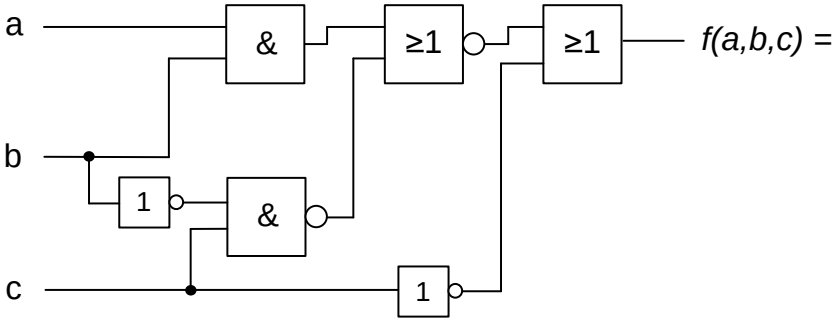
Lösung: In diesem Fall muss man für alle beliebigen Probleme $X \in NP$ zeigen, dass eine Polynomialzeitreduktion auf SAT existiert, also $X \leq_{pol} SAT$. Cook hat das geschafft, indem er eine Funktion angegeben hat, die aus einer beliebigen nichtdeterministischen polynomialzeitbeschränkten Turingmaschine in Polynomialzeit eine aussagenlogische Formel generiert, sodass die Turingmaschine genau dann akzeptierend hält, wenn die Formel erfüllbar ist. Da es für alle Probleme $X \in NP$ so eine Turingmaschine gibt und der Beweis für alle solchen Turingmaschinen gilt, ist dadurch gezeigt, dass $\forall X \in NP : X \leq_{pol} SAT$, was die Definition von NP-Schwere für das Problem SAT ist.

Aufgabe 6 **10 Punkte**

2014-H-06 **Schaltwerke / CMOS**

/ 10

Das folgende Schaltnetz soll auf physikalischer Ebene in CMOS realisiert werden. Ihnen als Ingenieur stehen dazu maximal 8 PMOS und 8 NMOS Transistoren zur Verfügung. Zeichnen Sie von dem Schaltnetz auf Transistorebene nur den PMOS-Bereich. Betrachten Sie das NMOS-Rechteck als Blackbox und stellen Sie durch geeignete Leitungen eine Verbindung zwischen NMOS- und PMOS-Bereich her.



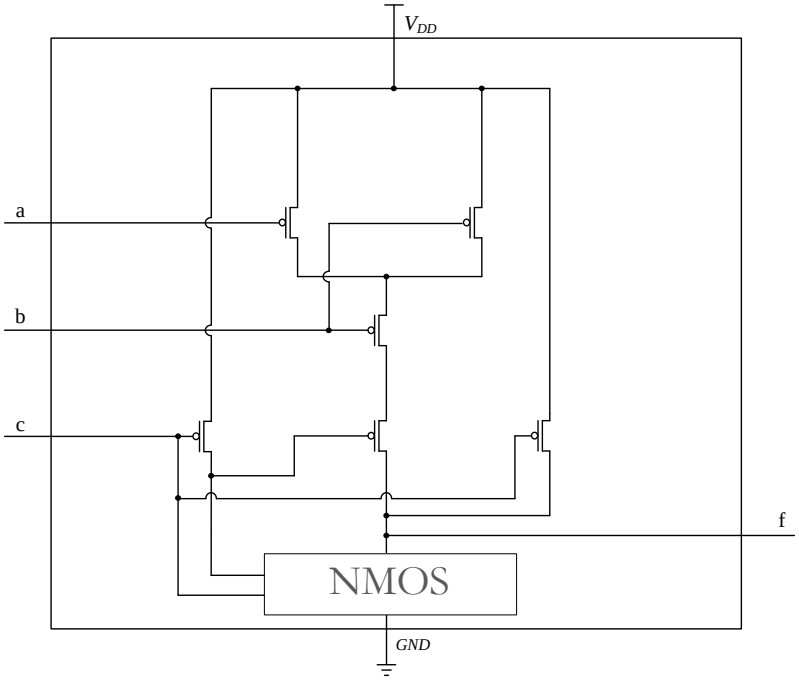
Hinweis: Lesen Sie zuerst die Boolesche Funktion aus dem Schaltnetz aus und vereinfachen Sie diese soweit wie möglich.

Lösung 1:

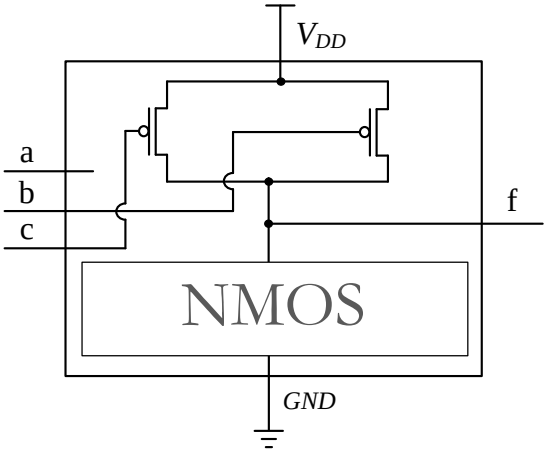
$f(a,b,c) = \neg c \vee ((a \wedge b) \overline{\vee} (\neg b \overline{\wedge} c))$

$= \neg c \vee (\neg(a \wedge b) \wedge (\neg b \wedge c))$

$= \neg c \vee ((\neg a \vee \neg b) \wedge (\neg b \wedge c))$



Lösung 2:
f kann vereinfacht werden zu
$a \overline{\wedge} b = \neg b \vee \neg c$



Aufgabe 7 **8 Punkte**

2014-H-07 **Huffman-Kodierung**

/ 8

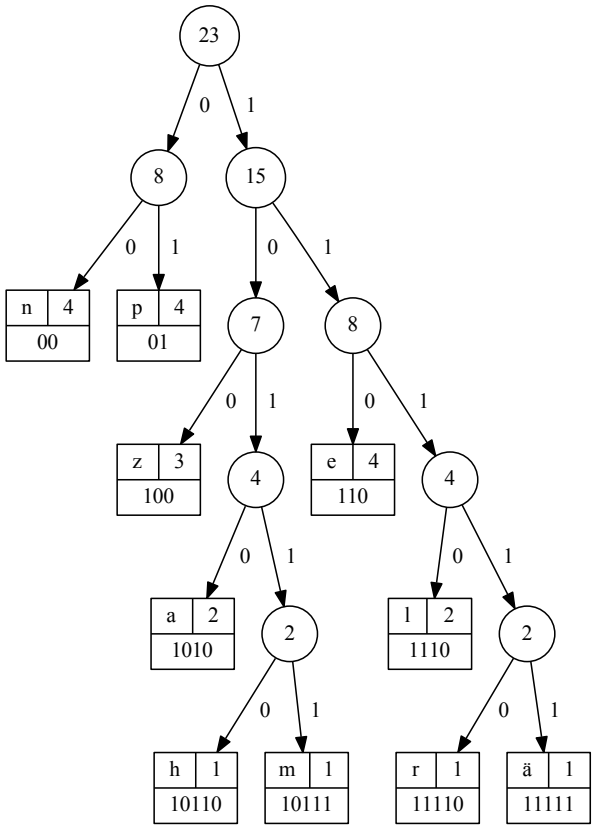
Folgende Zeichenkette der Länge 23 sei repräsentativ für Daten, die noch kommen sollen (Leerzeichen werden ignoriert):

ZEHN ZAPPELMÄNNER ZAPPELN

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	Z	E	H	N	A	P	L	M	Ä	R	
Häufigkeit	3	4	1	4	2	4	2	1	1	1	

Lösung (Beispiel):



Aufgabe 8 **9 Punkte**

2014-H-08 **Binary Decision Diagram**

/ 9

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

$$f(a,b,c) = \neg abc + a\neg bc + ab\neg c + \neg a\neg b\neg c = \neg(a \oplus b \oplus c)$$

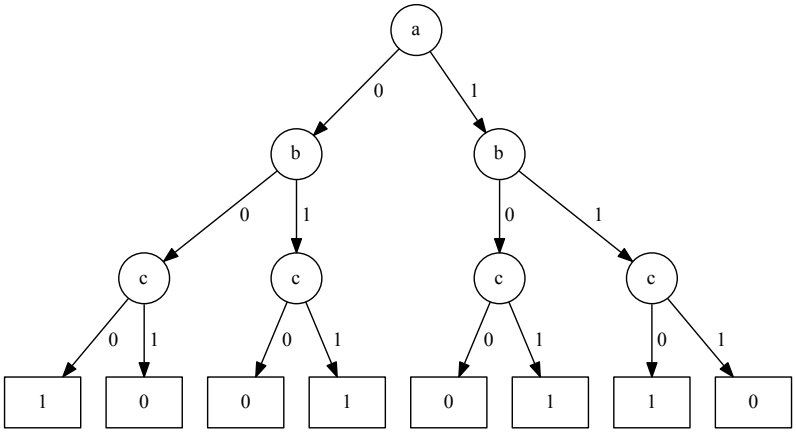
(a) Tragen Sie in die folgende Tabelle die Funktionswerte der Funktion f ein.

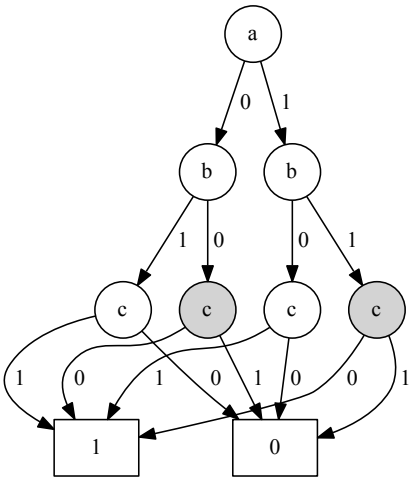
/ 2

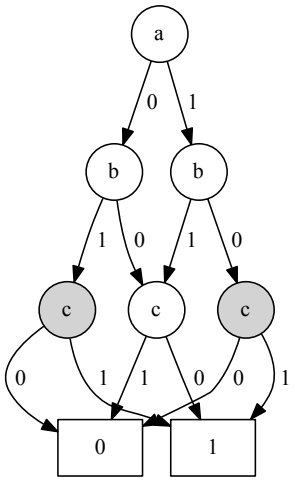
a	b	c	$a \oplus b \oplus c$	f
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

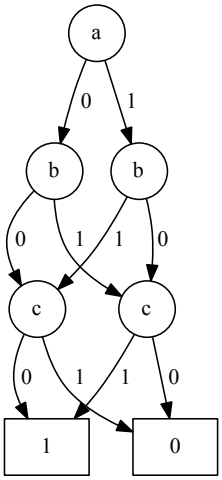
(b) Berechnen Sie das zu f gehörende BDD mit der Variablenreihenfolge $a \to b \to c$, indem Sie mit einer Baumdarstellung der Wertetabelle beginnen und den Algorithmus aus der Vorlesung anwenden.

/ 6




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

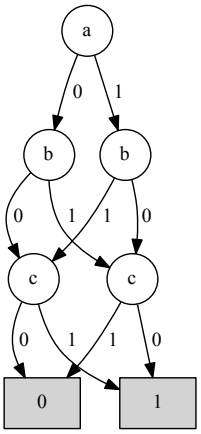

a
0
1
b
b
1
0
1
0
c
c
c
0
1
1
0
0
1
0
1


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

(c) Wie lässt sich aus einem gegebenen BDD für eine Funktion f das BDD für die negierte Funktion $f' =_{def} \neg f$ generieren? Erklären Sie in kurzen Worten.

/ 1

Lösung: Blattknoten 0 und 1 vertauschen.



Aufgabe 9	**7 Punkte**
2014-H-09	**Zahlendarstellung**

/ 7

Gegben sei folgende Gleitpunktzahl z in der IEEE-754-Darstellung.

10100001011010000100000000000000

(a) Geben Sie z im Dezimalsystem an. Es ist ausreichend, die Mantisse als Summe von Brüchen zu schreiben.

/ 2

Lösung:
Bei 32-Bit Gleitpunktzahlen im IEEE-Standard 754 gilt:
1-Bit Vorzeichen, 8-Bit Charakteristik, 23-Bit Mantisse.
$q = 2^{8-1} - 1 = 127$
$c = 2^6 + 2^1 = 66$
$m' = 2^{-1} + 2^{-2} + 2^{-4} + 2^{-9}$
$v = 1$
$z = -(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16} + \frac{1}{512}) \times 2^{-61}$

(b) Geben Sie z so genau wie möglich als 16-Bit Gleitpunktzahl z' an, mit folgender Aufteilung der Bits:

- Vorzeichen: 1 Bit,
- Charakteristik: 7 Bit,
- Mantisse: 8 Bit.

Wie groß ist der Fehler und wodurch wird er verursacht?

/ 3

Lösung:
$v' = 1$
$q' = 2^{7-1} - 1 = 63$
$c' - q' = -61$
$c' = q' - 61 = 2^1 = (0000010)_2$
Mantisse wird übernommen, 8 Bit sind aber nicht ausreichend
$\Rightarrow 2^{-9}$ kann nicht dargestellt werden, was zu einem Fehler von $2^{-9} \times 2^{-61} = 2^{-70}$ führt.
$m'' = (11010000)_2$

(c) Könnte z mit einer anderen Aufteilung der 16 Bit in Vorzeichen, Charakteristik und Mantisse genauer dargestellt werden? Begründen Sie Ihre Antwort.

/ 2

Lösung:
Nein. Das Fehlende Bit der Mantisse müsste entweder von Vorzeichen oder Charakteristik genommen werden. Das Vorzeichen-Bit wird benötigt zur Darstellung der Negativität und eine 6-Bit Charakteristik kann nur Exponenten von -30 bis 31 darstellen, also nicht -61.

Aufgabe 10 **5 Punkte**

2014-H-10 **Assembler**

/ 5

Gegeben seien folgende beiden Auszüge aus Assembler-Programmen `A1` und `A2`:

```
A1:
STORE R1 R3
STORE R2 R1
STORE R3 R2
```

```
A2:
ADD R1 R2 R1
SUBTRACT R1 R2 R2
SUBTRACT R1 R2 R1
```

(a) Geben Sie für die folgenden Anfangswerte von `R1` und `R2` an, welche Werte dort nach der Programmausführung von `A1` und `A2` jeweils gespeichert sind.

/ 3

	Werte vor Ausführung	Werte nach `A1`	Werte nach `A2`
R1	5	7	7
R2	7	5	5
R1	0	12	12
R2	12	0	0
R1	8	5	5
R2	5	8	8

(b) Welche Funktion berechnen `A1` und `A2` jeweils? Sind die beiden Programme äquivalent?

/ 2

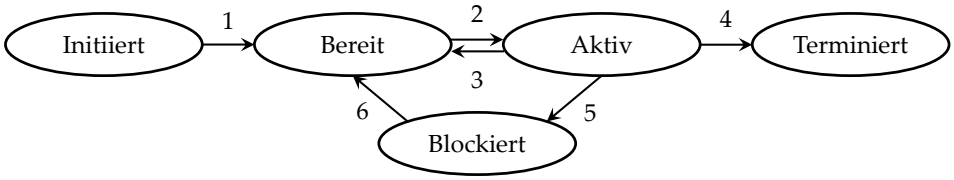
Lösung: Beide Programme vertauschen die Werte von R1 und R2, sind also äquivalent.

Aufgabe 11	**6 Punkte**
2014-H-11	**Betriebssysteme**

/ 6

(a) Beschreiben Sie in kurzen Worten die verschiedenen Zustandsübergänge eines Prozesses, wie sie in der folgenden Abbildung verdeutlicht sind. Verwenden Sie hierfür die Nummerierung aus der Abbildung.

/ 3



Lösung:

(1) Es sind alle Betriebsmittel außer dem Prozessor vorhanden, der Prozess ist bereit zur Ausführung

(2) Prozessor wird dem Prozess zugeteilt

(3) aktiver Prozess wird unterbrochen (Prozessor wird entzogen), Prozess bleibt aber bereit zur Fortsetzung der Ausführung

(4) Prozess ist abgeschlossen und gibt alle Betriebsmittel frei

(5) aktiver Prozess wird unterbrochen uns muss auf ein Ereignis warten

(6) Ereignis ist eingetreten, Prozess ist wieder bereit, wartet nur auf Prozessor

(b) In welchem Zusammenhang in Bezug auf Prozesse wird das Semaphore-Konzept verwendet? Was wird dadurch verhindert? Beschreiben Sie kurz die Idee dieses Konzeptes.

/ 3

Lösung:

Das Semaphore-Konzept wird bei der Synchronisation von Prozessen verwendet, um Prozesse zu koordinieren und zu verhindern, dass ein Prozess ein falsches Ergebnis liefert. Es ähnelt einer Verkehrsampel. Wenn ein Prozess einen kritischen Bereich betreten möchte, dann überprüft er zuerst, ob das Semaphore auf rot gesetzt ist (und wartet in diesem Fall, bis das Semaphore auf grün gesetzt wird). Betritt ein Prozess einen kritischen Bereich, dann schaltet das Semaphore auf rot. Verlässt er den Bereich wieder, dann schaltet es wieder auf grün.