Lösung

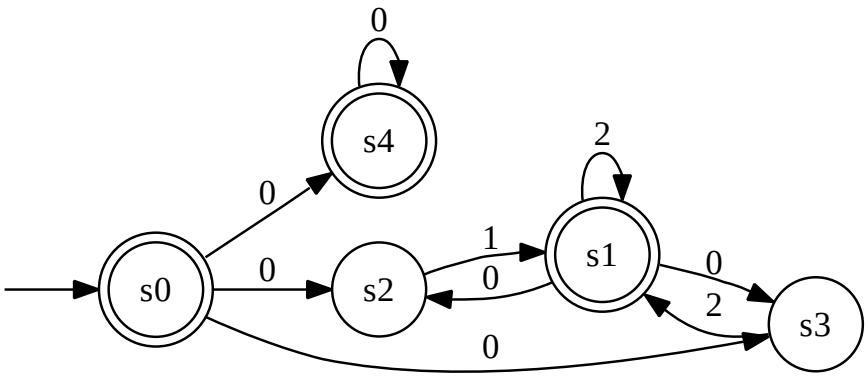
(90 Punkte)

Aufgabenübersicht

Aufgabe 1. Endliche Automaten (4 + 6 = 10 Punkte)

Gegeben sei der nichtdeterministische endliche Automat A:

$$A = (\{0, 1, 2\}, \{s0, s1, s2, s3, s4\}, \delta, s0, \{s0, s1, s4\}),$$
$$\delta :$$



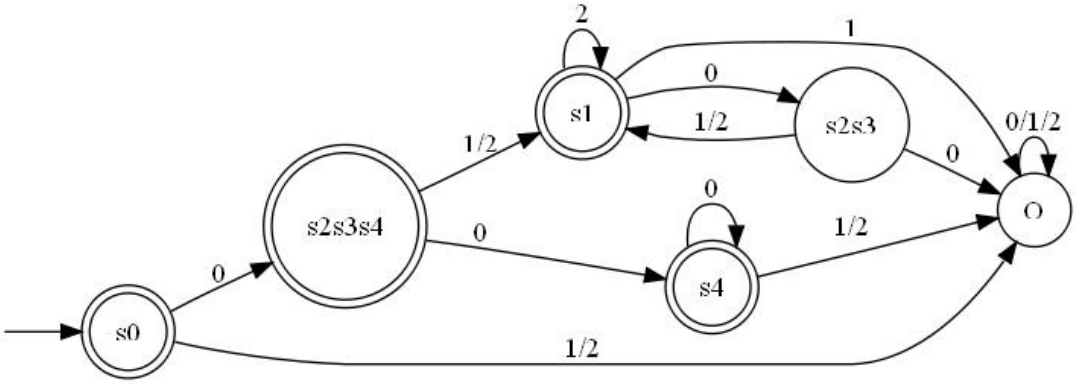
(a) Geben Sie einen regulären Ausdruck α an, sodass gilt: $L(A) = L(\alpha)$.

Lösung: $\alpha = (01 + 02)(01 + 02 + 2)^\star + 0^\star$. Dabei gilt: $x^+ := xx^\star$.

(b) Erzeugen Sie mithilfe des aus der Vorlesung bekannten Algorithmus einen deterministischen endlichen Automaten $A' = (E, S', \delta', s0', F')$, für den gilt: $L(A') = L(A)$. Nutzen Sie dafür die vorgegebene Tabelle. Definieren Sie A' vollständig; zeichnen Sie insbesondere ein Zustandsüberführungsdiagramm.

δ'	0	1	2
$\{s0\}$	$\{s2, s3, s4\}$	$\emptyset$	$\emptyset$
$\{s1\}$	$\{s2, s3\}$	$\emptyset$	$\{s1\}$
$\{s4\}$	$\{s4\}$	$\emptyset$	$\emptyset$
$\{s2, s3, s4\}$	$\{s4\}$	$\{s1\}$	$\{s1\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\{s2, s3\}$	$\emptyset$	$\{s1\}$	$\{s1\}$

$A' = (\{0, 1, 2\}, \{s0, s1, s4, s2s3, s2s3s4, O\}, \delta', s_0, \{s2s3s4, s0, s1, s4\})$:



Aufgabe 2. Grammatiken, Chomsky-Normalform (1 + 4 + 5 + 1 = 11 Punkte)

Gegeben sei die Grammatik

$$G = (\{S, X, Y, Z\}, \{1, 2, 3\}, P, S)$$

mit

$$\begin{aligned} P = \{S &\to XYZ, \\ X &\to 1X \mid 1, \\ Y &\to 22Y \mid 22, \\ Z &\to 333Z \mid 333\} \end{aligned}$$

(a) Geben Sie das größte i an, für das gilt: G ist vom Chomsky-Typ i.

Lösung: Typ 2.

(b) Geben Sie das größte j an, für das gilt: $L(G)$ ist vom Chomsky-Typ j; beweisen Sie die Richtigkeit Ihrer Wahl entweder durch Angabe einer äquivalenten Grammatik des entsprechenden Typs oder der Anwendung des Pumping-Lemmas.

Lösung: Typ 3.

$$G_{reg} = (\{X, Y_1, Y_2, Z_1, Z_2, Z_3\}, \{1, 2, 3\}, P_{reg}, X)$$

mit

$$\begin{aligned} P_{reg} = \{X_0 &\to 1X_1, \\ X_1 &\to 1X_1 \mid 2Y_1, \\ Y_1 &\to 2Y_2, \\ Y_2 &\to 2Y_1 \mid 3Z_1, \\ Z_1 &\to 3Z_2, \\ Z_2 &\to 3Z_3, \\ Z_3 &\to 3Z_1 \mid \lambda\} \end{aligned}$$

(c) Erzeugen Sie aus G eine äquivalente Grammatik $G' = (N', T, P', S')$ in Chomsky-Normalform durch Anwendung des in der Vorlesung vorgestellten Algorithmus.

Lösung:

(1) Lambda-frei machen. ==> Entfällt.

(2) Elimiere Umbenennungen. ==> Entfällt.

(3) 1 Terminalsymbol oder beliebig viele Nonterminalesymbole auf rechter Seite.

$$
\begin{aligned}
S &\to XYZ \\
X &\to AX|1 \\
Y &\to BBY|BB \\
Z &\to CCCZ|CCC \\
A &\to 1 \\
B &\to 2 \\
C &\to 3
\end{aligned}
$$

(4) Genau zwei Nonterminalesymbole auf rechter Seite.

$$
\begin{aligned}
S &\to MZ \\
M &\to XY \\
X &\to AX|1 \\
Y &\to NY|BB \\
N &\to BB \\
Z &\to OP|OC \\
O &\to CC \\
P &\to CZ \\
A &\to 1 \\
B &\to 2 \\
C &\to 3
\end{aligned}
$$

(d) Was muss man tun, um eine rechtslineare Grammatik in die Greibach-Normalform zu bringen?

Lösung: Nichts.

Aufgabe 3. Kellerautomat (7 + 1 = 8 Punkte)

Gegeben sei für $E = \{a, b, c, d\}$ folgende Sprache:

$$L = \bigcup_{m \geq n \geq 0} \{a, b\}^m \{c, d\}^n$$

Es gilt beispielsweise: $aabccc, aa, abaacdc, \lambda \in L$; $adba, aaccc, d, acdc \notin L$.

(a) Geben Sie einen deterministischen Kellerautomaten $KA = (E, S, K, \delta, s_0, k_0, F)$ an mit $L(KA) = L$. Zeigen Sie, dass Ihr Kellerautomat das Testwort *ababcd* akzeptiert.

Hinweis: Der Keller muss zum Akzeptieren nicht unbedingt leer sein.

Lösung:

$$KA = (\{a, b, c, d\}, \{s_0, s_1\}, \{k_0, a\}, \delta, s_0, k_0, \{s_0, s_1\}).$$

δ :

$$\delta(s_0, a, k_0) = (s_0, ak_0)$$
$$\delta(s_0, b, k_0) = (s_0, ak_0)$$
$$\delta(s_0, a, a) = (s_0, aa)$$
$$\delta(s_0, b, a) = (s_0, aa)$$
$$\delta(s_0, c, a) = (s_1, \lambda)$$
$$\delta(s_0, d, a) = (s_1, \lambda)$$
$$\delta(s_1, c, a) = (s_1, \lambda)$$
$$\delta(s_1, d, a) = (s_1, \lambda)$$

Erkennung des Testwortes *ababcd*:

$$(s_0, ababcd, k_0) \vdash (s_0, babcd, ak_0) \vdash (s_0, abcd, aak_0) \vdash (s_0, bcd, aaak_0) \vdash$$
$$(s_0, cd, aaaak_0) \vdash (s_1, d, aaak_0) \vdash (s_1, \lambda, aak_0)$$

(b) Wie unterscheidet sich das Modell des deterministischen Kellerautomaten vom Modell des nichtdeterministischen Kellerautomaten bezogen auf die folgenden Eigenschaften (in jeder Zeile ist genau eine Antwort richtig): **Lösung:**

Sprachmächtigkeit	gleich	⊠ ungleich	unbekannt
Eignung für Programmiersprachen	gleich	⊠ ungleich	unbekannt

Aufgabe 4. Turingmaschine (5 + 2 = 7 Punkte)

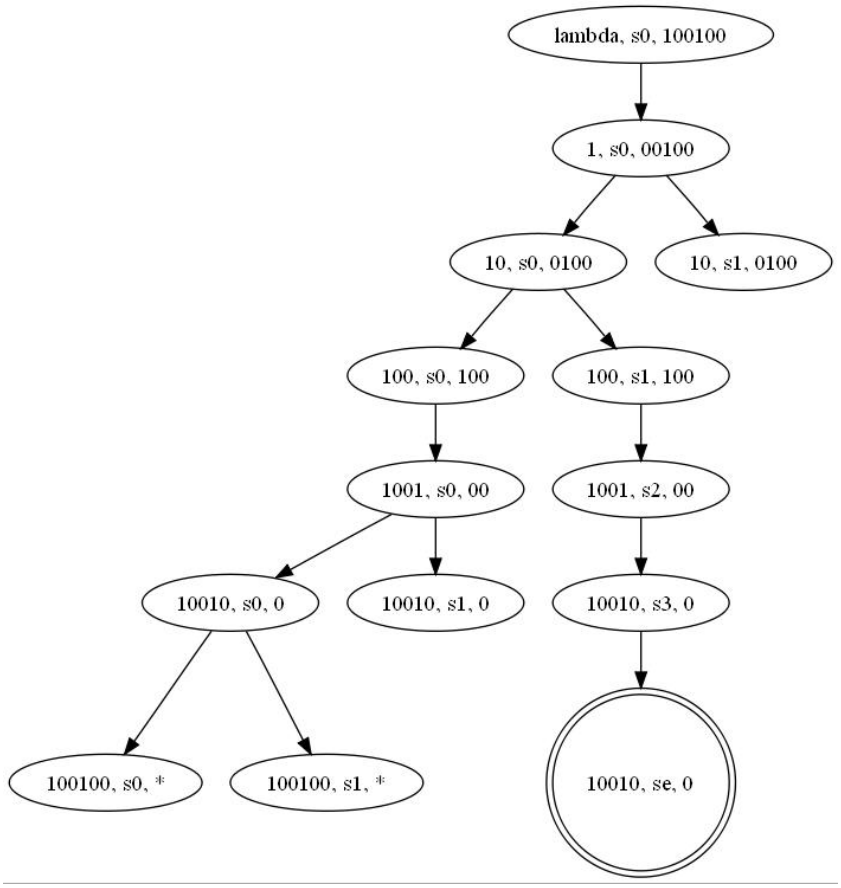
Gegeben sei die nichtdeterministische Turingmaschine

$$T = (\{0,1\}, \{0,1,\star\}, \{s_0, s_1, s_2, s_3, s_e\}, \delta, s_0, \{s_e\}).$$

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

(a) Geben Sie für das Wort $w = 100100$ den Konfigurationsbaum an, der von T' auf Eingabe von w abgearbeitet wird. Markieren Sie eventuell vorkommende Endkonfigurationen.

Lösung:



(b) Geben Sie die von T akzeptierte Sprache $L(T)$ an. Definieren Sie $L(T)$ mathematisch oder formulieren Sie umgangssprachlich präzise.

Lösung:

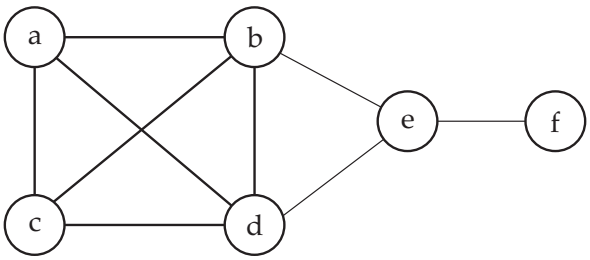
Jedes Wort über $E = \{0, 1\}$, das als Teilwort das Wort 0100 enthält, d. h.

$$L(T) = \{0, 1\}^\star 0100 \{0, 1\}^\star.$$

Aufgabe 5. Komplexitätstheorie (3 + 7 = 10 Punkte)

In der Vorlesung wurde das *NP*-vollständige Problem *Clique* vorgestellt. Es bezeichnet die folgende Fragestellung: Sei $G = (V, E)$ ein (ungerichteter) Graph mit der Knotenmenge V und der Kantenmenge E; sei $k > 0$ eine natürliche Zahl; gibt es in G mindestens k Knoten, die paarweise durch eine Kante verbunden sind, also eine „Clique“ bilden?

Zum Beispiel bilden im abgebildeten Graphen die Knoten a, b, c und d die größte Clique mit $k = 4$.



Formal:

$$Clique = \{(G,k) \mid k \in \mathbb{N}, G = (V,E), \exists\, V' \subseteq V \text{ mit } |V'| = k : \ \forall\, v_1, v_2 \in V' : \ (v_1, v_2) \in E\}$$

(a) Es sei bereits bekannt, dass *Clique NP*-vollständig ist. Für ein konstantes $c \in \mathbb{N}$ sei folgendes modifizierte Clique-Problem $Clique_{mod-c}$ gegeben:

$$Clique_{mod-c} = \{(G,k) \mid k \geq c, (G, k-c) \in Clique\}$$

Durch Erhöhen von c kann man das Problem also „vereinfachen“, weil man nur noch um c kleinere Cliquen finden muss, während die Graphen unverändert bleiben.

Zeigen Sie, dass $Clique_{mod-c}$ ebenfalls *NP*-vollständig ist.

- $Clique_{mod-c} \in NP$:
- $Clique_{mod-c}$ $NP-$schwer:

Lösung:

- $Clique_{mod-c} \in NP$: Eine nichtdeterministische Turingmaschine kann eine Knoten-Teilmenge $V' \subseteq V$ mit $|V'| = k - c$ raten und in Polynomialzeit verifizieren, dass die Knoten eine Clique bilden.
- $Clique_{mod-c}$ ist *NP*-schwierig: Reduziere *Clique* auf $Clique_{mod-c}$. Sei (G, k) eine Eingabe für *Clique*. Die Funktion

$$f : Graph \times \mathbb{N} \to Graph \times \mathbb{N} : f(G,k) = (G, k+c)$$

ist in Polynomialzeit berechenbar und es gilt:

$$(G,k) \in Clique \Leftrightarrow f(G,k) \in Clique_{mod-c}.$$

Also kann man eine Eingabe für *Clique* in Pol.-Zeit in eine Eingabe für $Clique_{mod-c}$ umwandeln und dann das Ergebnis mit einem Algorithmus für $Clique_{mod-c}$ berechnen (eine Rückumwandlung der Ausgabe ist hier nicht erforderlich bzw. trivial). Wenn $Clique_{mod-c}$ also in Polynomialzeit berechenbar wäre, wäre es auch *Clique* und damit alle Probleme in *NP*.

(b) Gegeben sei nun für jedes feste $k \in \mathbb{N}$ folgendes Problem $Clique_{einfach-k}$:

$$Clique_{einfach-k} = \{G \mid (G, k) \in Clique\}.$$

Es ist also wieder für einen Graphen G gefragt, ob er eine Clique der Größe k besitzt, nur ist k nun nicht mehr Teil der Eingabe, sondern bleibt konstant.

Begründen Sie, dass für beliebige k das Problem $Clique_{einfach-k}$ in *P* liegt.

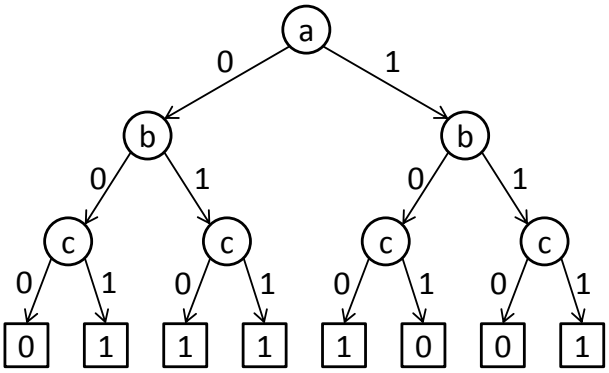
Hinweis: Überlegen Sie zunächst, wie die Situation für $k = 1$ oder $k = 2$ ist. **Lösung:** Hier muss man sich alle k-elementigen Teilmengen der n-elementigen Knotenmenge anschauen und kommt also auf

$$\begin{pmatrix} n \\ k \end{pmatrix} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\cdots 2}{k!(n-k)(n-k-1)(n-k-2)\cdots 2}$$
$$= \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!} \leq \frac{n^k}{k!} \in O(n^k)$$

Möglichkeiten. In jeder dieser k-elementigen Knotenteilmengen muss man nun alle Knoten paarweise überprüfen, um festzustellen, ob alle durch eine Kante verbunden sind. Das sind nochmal $O(k^2)$ Überprüfungen. Insgesamt ergibt sich also eine Laufzeit von $O(n^k \cdot k^2)$, die bei konstantem k polynomiell ist.

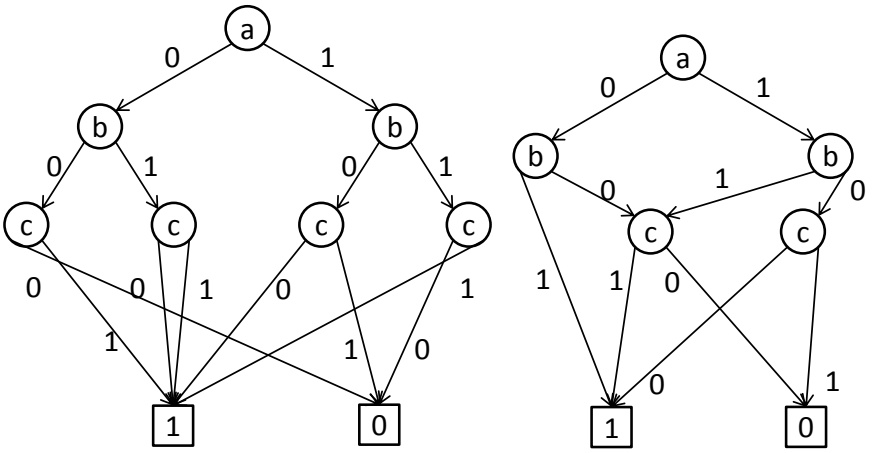
Aufgabe 6. Binary-Decision-Diagram (6 + 2 = 8 Punkte)

Gegeben sei die durch den abgebildeten Baum definierte Boolesche Funktion $f : \mathbb{B}^3 \to \mathbb{B}$.



(a) Erzeugen Sie das zu f gehörende BDD.

Lösung:

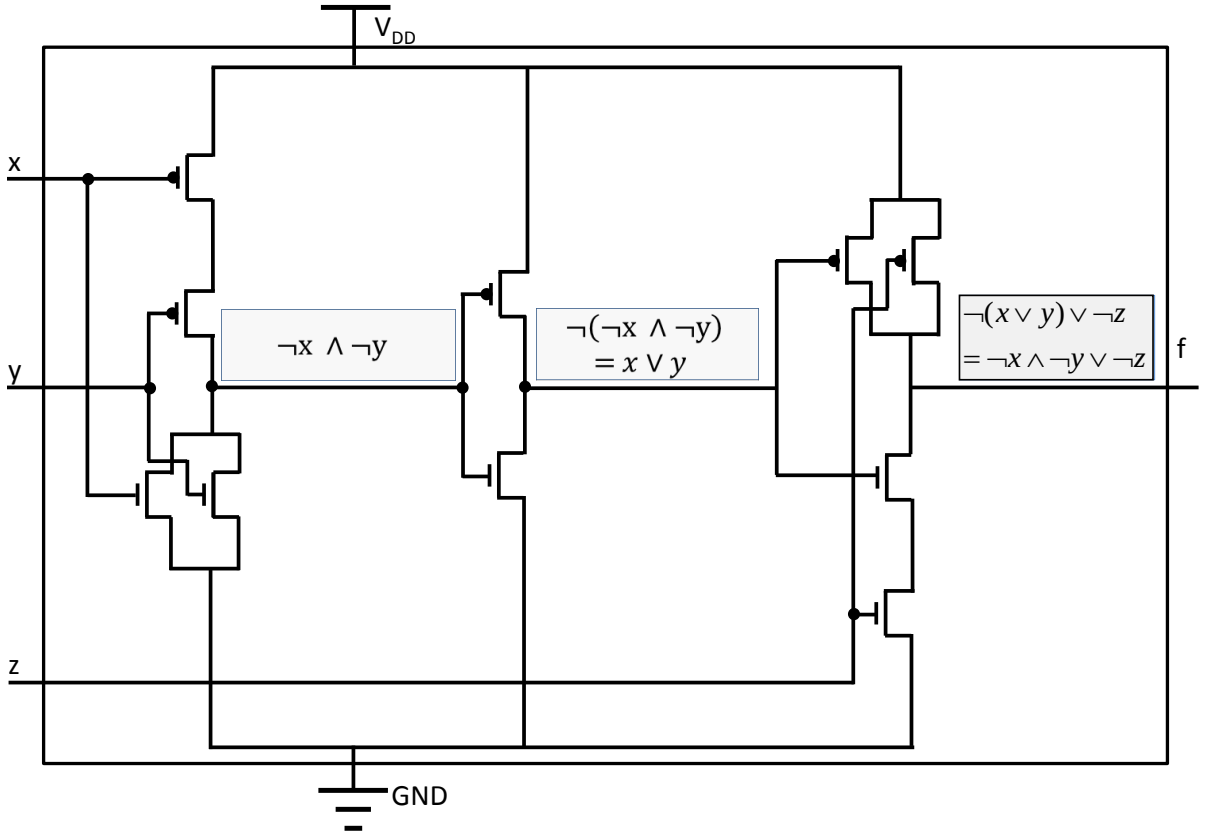


(b) Geben Sie die Funktion f als Booleschen Ausdruck an.

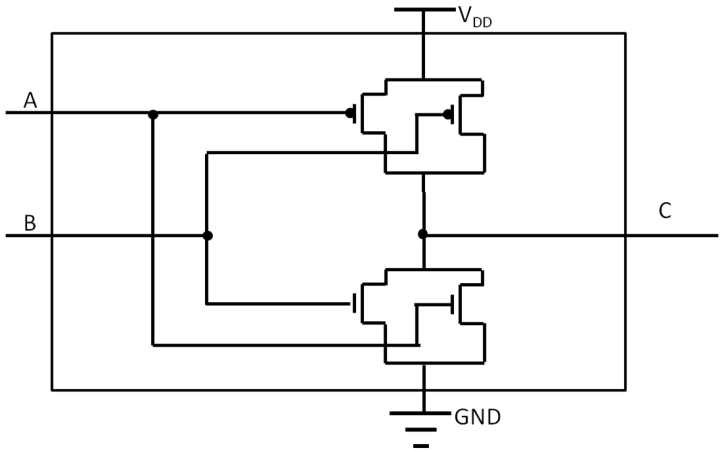
Lösung:
$f(a,b,c) = a' \cdot b + a' \cdot b' \cdot c + a \cdot b \cdot c + a \cdot b' \cdot c'$

Aufgabe 7. CMOS (6 + 2 = 8 Punkte)

(a) Gegeben sei die Funktion $f : \mathbb{B}^3 \to \mathbb{B}$, die durch die abgebildete CMOS-Schaltung definiert wird. Geben Sie die Funktion f als Term in Boolescher Algebra an. Schreiben Sie hierfür die Zwischenergebnisse in die grau markierten Felder.
Lösung:



(b) Gegeben sei nun die folgende aus PMOS- und NMOS-Bausteinen bestehende Schaltung:



Geben Sie an, was passiert, wenn für die Signale A, B Folgendes gilt:

- $A = B$ Bei $A = B$ funktioniert die Schaltung weiterhin als Inverter.
- $A \neq B$ $A \neq B$: In beiden Zweigen leitet ein Transistor und der andere sperrt. Wegen der Paralleschaltung gibt es einen geringen Widerstand zwischen VDD und GND, es fließt ein hoher Strom, der das Bauelement zerstören könnte. C liegt irgendwo zwischen VDD und GND.

Aufgabe 8. Huffman-Kodierung (7 + 1 = 8 Punkte)

Folgende Zeichenkette sei repräsentativ für Daten, die noch kommen sollen:

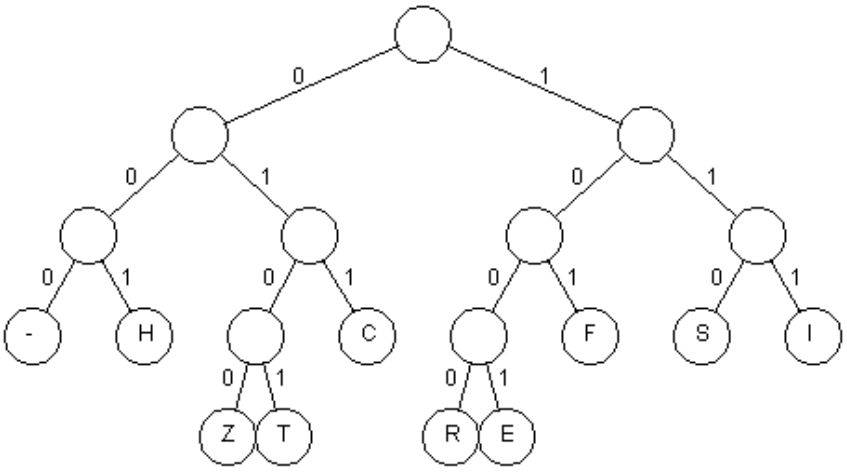
FISCHERS-FRITZ-FISCHT-FISCHE

(a) 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	F	I	S	H	-	C	E	R	T	Z
Häufigkeit	4	4	4	3	3	3	2	2	2	1

Zeichen	Code
F	101
I	111
S	110
H	001
-	000
C	011
E	1001
R	1000
T	0101
Z	0100



(b) Wie viele Bits / Zeichen spart man durch diese Kodierung ein gegenüber einer herkömmlichen Kodierung, die 4 Bits / Zeichen benötigt.

Lösung: Codelänge: $\frac{1}{28} \cdot (3 \cdot 4 \cdot 3 + 3 \cdot 3 \cdot 3 + 3 \cdot 2 \cdot 4 + 1 \cdot 1 \cdot 4) = \frac{91}{28} = 3,25$ Bits / Z. Daraus folgt, dass man $\frac{3}{4}$ Bits / Z. spart.

Aufgabe 9. TCP, IP, UDP (2 + 4 = 6 Punkte)

Für die Kommunikation im Internet werden verschiedene Protokolle verwendet. Zu diesen Protokollen gehören unter anderem TCP, IP und UDP.

(a) Geben Sie an, wofür die Abkürzungen TCP, IP und UDP stehen.
Lösung:
TCP: Transmission Control Protocol
IP: Internet Protocol
UDP: User Datagram Protocol

(b) Diesen drei Protokollen kommen unterschiedliche Aufgaben bei der Kommunikation im Internet zu.

Geben Sie in der zweiten Spalte der folgenden Tabelle an, welche der drei genannten Protokolle daran beteiligt sind. Sie können einer Aufgabe ein oder mehrere Protokolle zuweisen.

Lösung:

Adressierung und Versendung von Datenpaketen	IP
Unterteilung des Datenstroms zwischen Sender und Empfänger in Pakete	TCP / UDP
Datensicherheit und Datenflusssteuerung	TCP
Verwendung des „Open Shortest Path First“-Verfahrens	IP
MultiMedia-Übertragungen	UDP
Herstellung der Verbindung zwischen Sender und Empfänger	TCP
Auswahl des nächsten Knotens auf dem Weg von Quelle zu Ziel	IP

Aufgabe 10. Assembler (5 + 1 = 6 Punkte)

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, R3
2. L1: MULTIPLY R1, R2, R2
3.      MULTIPLY R2, R1, R2
4.      SUBTRACT R3, #1, R3
5.      JNZ R3, L1
6.      ADD R1, R2, R1
```

(a) Am Anfang der Rechnung stehen in den Registern $R1$ und $R2$ die Werte $R1^A = x \in \mathbb{N}^+$ und $R2^A = y \in \mathbb{N}^+$. Wie hängt der Endwert $R1^E := f(x, y)$ des Registers $R1$ am Ende der Rechnung von x, y ab, welche Funktion $f(x, y)$ berechnet das Programm also?

(b) Wird die Funktion $f(x, y)$ auch richtig berechnet, wenn zu Beginn der Wert $R1^A = x = 0$ in $R1$ steht? Begründen Sie kurz.

Lösung: (a) Es wird für $x, y > 0$ die Funktion $f(x, y) = x^{2x} \cdot y + x$ berechnet.

(b) für $x = 0$ wird die Funktion nicht korrekt berechnet, es kommt nämlich zu einer Endlosschleife, da $R3$ in Zeile 5 nie 0 ist.

Aufgabe 11. Dateiorganisation (4 + 2 + 2 = 8 Punkte)

Eine Möglichkeit der Organisation von Dateien ist die sogenannte Hash-Organisation. Es sei folgende Hash-Funktion $h : \mathbb{N} \to \mathbb{N}$ gegeben:

$$h(s) = s \bmod 10$$

(a) Eine Bank speichert ihre gesamten Kundendatensätze mit Hilfe dieser Hash-Funktion. Dabei verwendet diese die Kontonummer als Primärschlüssel s. Speichern Sie folgende Kontonummern in der angegebenen Datenbank:

$$154636, 456255, 992348, 122222, 555555, 246799, 111001, 667761.$$

Verwenden Sie hierbei zur Kollisionsbehandlung lineares Austesten auf Satzebene.
Lösung:

relative Satznummer	Satz mit Schlüssel
0	
1	111001
2	122222
3	667761
4	
5	456255
6	154636
7	555555
8	992348
9	246799

(b) Welche weiteren Möglichkeiten zur Vermeidung von Kollisionen kennen Sie? Nennen Sie hiervon zwei.
Lösung:
Vermeidung von Kollisionen durch lineares Austesten, Überlaufbereich, 2. Hash-Funktion, ...
Hash-Funktion muss eindeutig sein, z.B. $h(PS) = \sqrt{PS}$

(c) Begründen Sie, warum man diese Hash-Funktion in der Praxis nicht verwenden würde und geben Sie eine entsprechend verbesserte Hashfunktion an.

Lösung: Statt $M = 10$, M so wählen, dass Schlüssel gleichmäßig verteilt werden: M geeignet gewählte Primzahl, bspw. 13.