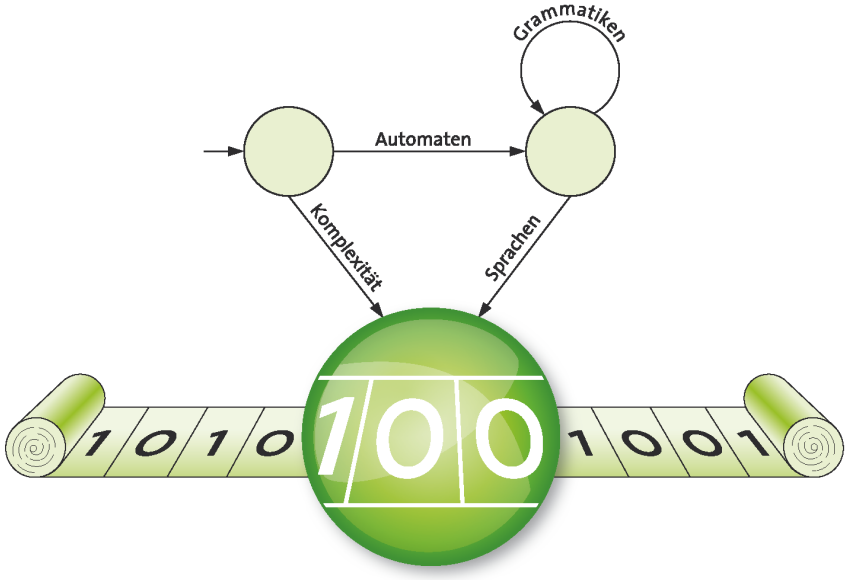
Oldenbourg Verlag

Lukas König, Friederike Pfeiffer-Bohnen,
Hartmut Schmeck

100 Übungsaufgaben zu Grundlagen der Informatik

Band I – Theoretische Informatik



F05

Liebe Studierende,

wir stellen Ihnen durch den vorliegenden Aufgabenpool eine Fülle an Übungsmaterial zur Verfügung, um Sie bei Ihrer Klausurvorbereitung zu unterstützen. Der Aufgabenpool ist auch als **Buch in zwei Bänden** erschienen:

`http://www.dasinfobuch.de/#uebung`

Zusätzlich zu den Aufgaben bieten die beiden Buchbände **kompakte Einführungen** in jedes Themenfeld. Sie sind auch als eBooks verfügbar, die nur über das KIT-Netz abgerufen werden können: Band I, Band II. Band I ist dabei auch als Übungsbuch zum Lehrbuch

Theoretische Informatik – ganz praktisch

ausgerichtet und wird von diesem aus referenziert.

Zweck des Aufgabenpools

Der Pool ist zum Üben der Inhalte der Vorlesung gedacht und geht dabei über die in den Tutorien vorgestellten Aufgaben hinaus. Die Bearbeitung der Aufgaben soll Ihnen helfen, ein besseres Gefühl für mögliche Fragestellungen zu bekommen, die sich aus den verschiedenen Themenbereichen ergeben. Die Aufgaben in der Klausur können sich natürlich trotzdem von den Poolaufgaben unterscheiden, sodass ein gründliches Verständnis des Stoffs zu einer erfolgreichen Klausurvorbereitung gehört. Die zusammenfassenden Einführungen in der Buchversion können Sie nutzen, um Ihr theoretisches Wissen zu jedem Kapitel aufzufrischen.

Aufbau und Diskussionsforen

Bei jeder Aufgabe des Aufgabenpools steht neben der Überschrift eine Einschätzung ihres Schwierigkeitsgrads. Dabei stehen die Symbole $\star$, $\star^\star$, $\star^{\star^\star}$ und $\star^{\star^{\star^\star}}$ für "leicht", "mittel", "schwer" und "sehr schwer". Links neben dem Aufgabentitel steht die Aufgaben-ID, die auch einen **anklickbaren Link der Form**

END-AA

darstellt. Darüber können Sie eine **Question/Answer-Plattform** (Q/A) erreichen, wo Lösungen und Probleme zu dieser Aufgabe diskutiert werden. Über den Link landen Sie in einer sogenannten "Kategorie" der Q/A-Seite, die alle Fragen zu dieser Aufgabe enthält. Sie können dort die Aufgaben auch bewerten und uns damit helfen, den Pool weiter zu verbessern.

Auch zu den Übungs- und Altklausuraufgaben gibt es auf der Q/A-Plattform Kategorien, die sie ebenfalls durch Anklicken der jeweiligen Aufgaben-ID erreichen können.

Qualität der Aufgaben

Wir haben die Aufgaben gründlich durchgesehen. Sollten Sie trotzdem Fehler, Unverständliches oder Ungenauigkeiten finden, weisen Sie uns bitte über die Q/A-Plattform darauf hin.

Möge der Pool Sie zu einem besseren Verständnis der Vorlesungsinhalte führen! Wir wünschen Ihnen viel Spaß beim Lernen, Verstehen und Diskutieren.

Ihr Info-II-Team
Karlsruhe, February 9, 2017

Contents

1 Endliche Automaten mit Ausgabe

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=242

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

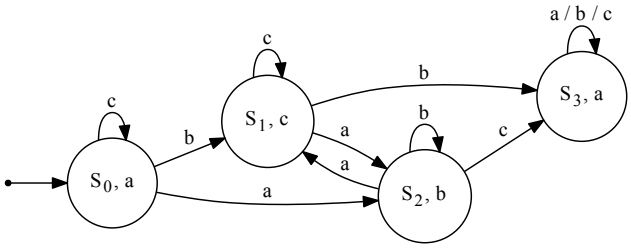
Aufgabe 1 ★

END-AU **Umwandlung von Moore-Automaten in Mealy-Automaten**

Gegeben sei folgender Moore-Automat M.

$$M = (\{a, b, c\}, \{s_0, s_1, s_2, s_3\}, \{a, b, c\}, \delta, \gamma, s_0)$$

δ, γ:



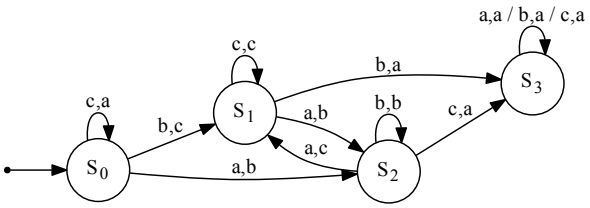
Wandeln Sie M in einen äquivalenten Mealy-Automaten M' um und geben Sie diesen vollständig an.

Lösung:

Zu M ergibt sich folgender Moore-Automat M'

$$M' = (\{a, b, c\}, \{s_0, s_1, s_2, s_3\}, \{a, b, c\}, \delta', \gamma', s_0)$$

δ', γ':



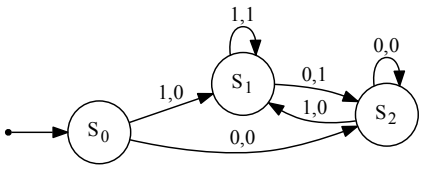
Aufgabe 2 ★⋆

END-AT **Mealy- und Moore-Automaten**

Gegeben sei der folgende Mealy-Automat $M = (E, S, A, \delta, \gamma, s_0)$.

$$M = (\{0,1\}, \{s_0, s_1, s_2\}, \{0,1\}, \delta, \gamma, s_0)$$

δ, γ:



a) Welche Funktion $f : E^\star \to A^\star$ berechnet M?

Lösung:

Der Automat M teilt gegebene Binärzahlen durch zwei, wobei ein gegebenenfalls auftretender Rest nicht ausgegeben wird. Der Automat bewirkt also eine Shift-Right-Operation. So wird beispielsweise aus der Eingabe der Binärzahl 10110011 die Ausgabe 01011001.

$$f(w) = \begin{cases} 0v, & w = vu \ (v \in E^\star, u \in E) \\ \lambda, & w = \lambda \end{cases}$$

b) Was ist der Unterschied zwischen Mealy- und Moore-Automaten?

Lösung:

Bei Mealy-Automaten wird die Ausgabe mit dem Zustandsübergang assoziiert, wohingegen bei Moore-Automaten die Ausgabe mit einem Zustand assoziiert wird.

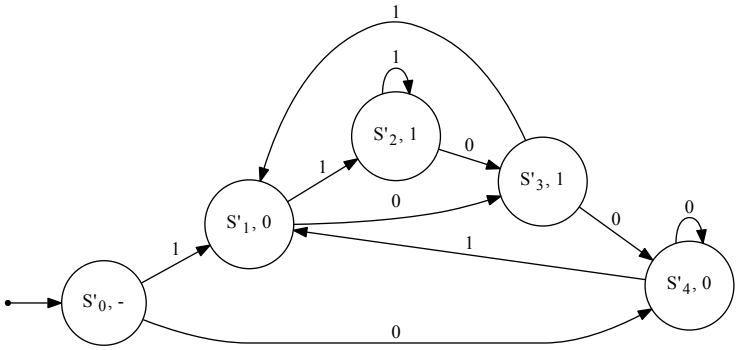
c) Geben Sie zu dem Mealy-Automaten M den zugehörigen Moore-Automaten M' an. Definieren Sie den M' vollständig.

Lösung:

Es ergibt sich folgender Moore-Automat M'

$$M' = (\{0,1\}, \{s'_0, s'_1, s'_2, s'_3, s'_4\}, \{-,0,1\}, \delta', \gamma', s'_0)$$

δ', γ':



Anmerkung: Die Ausgabe − im Startzustand wird für das Ergebnis der Teilung der Binärzahl ignoriert.

d) Warum kommen Mealy-Automaten gegenüber Moore-Automaten bei gleicher Funktionsweise mit weniger Zuständen aus?

Lösung:

Mealy-Automaten können mit einem Zustand für jedes Eingabesymbol eine andere Ausgabe erzeugen. Moore-Automaten brauchen dagegen für unterschiedliche Ausgaben auch unterschiedliche Zustände. Im schlechtesten Fall bräuchte man $S \times E$ als Zustandsmenge im Moore-Automaten.

Aufgabe 3 ★⋆

END-AA **Mealy-Automaten**

Entwerfen Sie einen Mealy-Automaten M zur bitseriellen Subtraktion zweier gleichlanger Zahlen. Geben Sie M vollständig an.

Lösung:

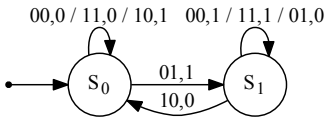
Folgende Wahrheitstabelle zeigt die bitserielle Subtraktion zweier Zahlen. Dabei ist a das entsprechende Bit des Minuenden, b das entsprechende Bit des Subtrahenden und c der jeweilige Übertrag. Als Ergebnis erhält man das neue Ergebnis-Bit d sowie den neuen Übertrag c'.

a	b	c	d	c'
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Somit ergibt sich folgender Mealy-Automat M:

$$M = (\{00, 01, 10, 11\}, \{s_0, s_1\}, \{0, 1\}, \delta, \gamma, s_0)$$

δ, γ:



Hierbei kommt den Zuständen folgende Bedeutung zu:

s_0 : Übertrag c bzw. c' ist 0
s_1 : Übertrag c bzw. c' ist 1

Aufgabe 4 ★★

END-AR **Mealy-Automaten**

Zur Darstellung von Zahlen kann die BCD-Kodierung verwendet werden (vgl. Band 2, Kapitel 5). Diese Blockkodierung ist eine Tetradenkodierung, wobei jede Ziffer einer Dezimalzahl durch vier Bits mit der Stellenwertigkeit $2^3 = 8, 2^2 = 4, 2^1 = 2$ und $2^0 = 1$ dargestellt wird. Der BCD-Code 0110 1001 0000 0100 entspricht beispielsweise der Dezimalzahl 6904.
Definieren Sie einen Mealy-Automaten M, der bei Eingabe einer BCD kodierten Zahl die entsprechende Dezimalzahl ausgibt. Geben Sie M vollständig an.
Hinweis: Gehen Sie davon aus, dass es sich bei allen eingegebenen Binärzahlen um BCD-Kodierungen handelt, eine Überprüfung dessen also nicht notwendig ist. Der andernfalls notwendige Auffangzustand wird als implizit vorhanden angenommen.

Lösung:

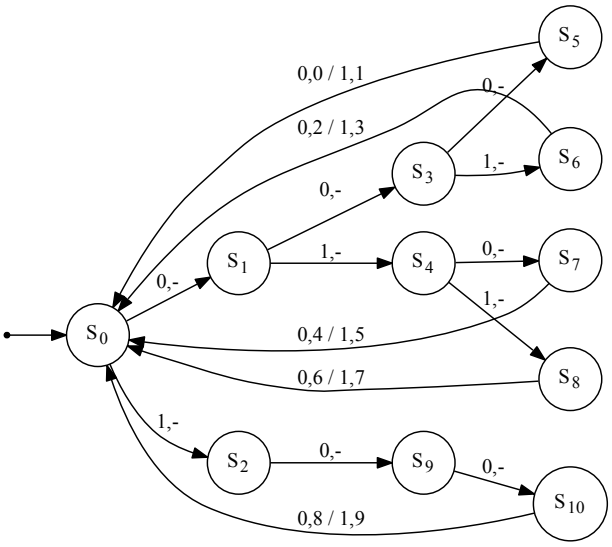
Die Zahlen 0 bis 9 werden bei der BCD-Kodierung folgendermaßen dargestellt:

0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Somit ergibt sich folgender Mealy-Automat M, wobei eine Ausgabe von $-$ ignoriert wird und somit erst nach der Eingabe einer BCD-Ziffer, also nach 4 Bits, eine interpretierbare Ausgabe erfolgt.

$$M(\{0,1\}, \{s_0, s_1, s_2\, s_3, s_4\, s_5, s_6, s_7\, s_8, s_9, s_{10}\}, \{-,0,1,2,3,4,5,6,7,8,9\}, \delta, \gamma, s_0)$$

δ, γ:



Aufgabe 5 ★⋆

END-AS **Mealy-Automaten**

Auf den meisten Golfplätzen findet man im Übungsbereich eine Driving Range, auf der lange Schläge geübt werden können. Die einzelnen Abschlagplätze sind nebeneinander aufgereiht und die Trainierenden schlagen aus Sicherheitsgründen alle in die gleiche Richtung. Die Golfbälle zum Trainieren kann man sich an speziellen Ballautomaten ausgeben lassen.
Konstruieren Sie den im folgenden Abschnitt beschriebenen Ballautomaten als Mealy-Automat $M = (E, S, A, \delta, \gamma, s_0)$. Geben Sie M vollständig an.

> Der zu modellierende Automat nimmt ausschließlich 1 Euro und 2 Euro Münzen an, alle anderen Münzen sind ungültig und werden zurückgegeben. Maximal wird vom Automaten eine Gesamteingabe von 3 Euro akzeptiert. Sofern mit einer weiteren eingeworfenen Münze dieser Betrag überschritten wird, wird die betreffende Münzen direkt zurückgegeben. Über eine Rückgabetaste kann der Kunde sich zudem das eingezahlte Geld auch direkt zurückgeben lassen, sollte er sich doch keine Golfbälle ausgeben lassen wollen. Kunden können zwischen drei verschiedenen Ballpaketen wählen und für jede dieser Optionen gibt es am Automaten eine Taste:
>
> 1) Die Ausgabe von 10 Golfbällen kostet 1 Euro,
>
> 2) 25 Golfbälle gibt es für 2 Euro und
>
> 3) 50 Golfbälle kosten 3 Euro.
>
> Bei einer Eingabe von beispielsweise 3 Euro kann der Kunde sich direkt 50 Golfbälle ausgeben lassen, es ist jedoch auch möglich, sich erst 25 Golfbälle ausgeben zu lassen und sich den Restbetrag von 1 Euro auszahlen zu lassen oder mit diesem Restbetrag 10 weitere Golfbälle zu ordern.

Lösung:

Eingabealphabet $E = \{M1, M2, MU, 10, 25, 50, MR\}$ mit der Bedeutung:

$$
\begin{aligned}
M1 &= \text{1 Euro Münze} \\
M2 &= \text{2 Euro Münze} \\
MU &= \text{Ungültige Münze} \\
10 &= \text{Taste zur Ausgabe von 10 Golfbällen} \\
25 &= \text{Taste zur Ausgabe von 25 Golfbällen} \\
50 &= \text{Taste zur Ausgabe von 50 Golfbällen} \\
MR &= \text{Rückgabetaste}
\end{aligned}
$$

Zustandsmenge $S = \{s_0, s_1, s_2, s_3\}$ 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

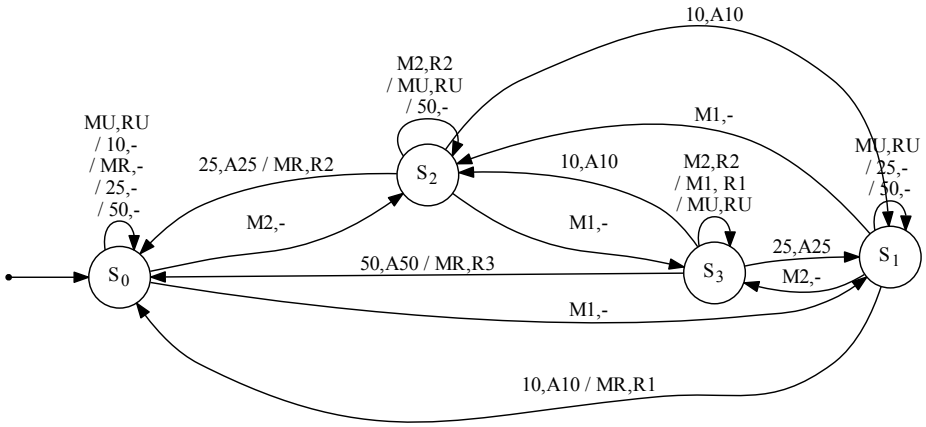
Ausgabealphabet $A = \{RU, R1, R2, R3, A10, A25, A50, -\}$ mit der Bedeutung:

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

Folglich ergibt sich folgender Mealy-Automat:

$M = (\{M1, M2, MU, 10, 25, 50, MR\}, \{s_0, s_1, s_2, s_3\}, \{RU, R1, R2, R3, A10, A25, A50, -\}, \delta, \gamma, s_0)$

δ, γ:



2 Endliche Automaten ohne Ausgabe

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=347

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 6 ★

END-AE **Endliche Automaten**

Gegeben sei die Sprache L mit

$$L = \{w \in \{0,1\}^\star \mid \exists u \in \{0,1\}^\star : w = 1u1\}$$

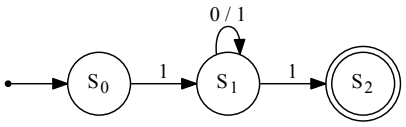
Die Sprache L enthält also alle Wörter über $\{0,1\}$, die mit einer 1 beginnen und mit einer 1 enden. Geben Sie einen nichtdeterministischen endlichen Automaten $A = (E, S, \delta, s_0, F)$ an mit $L(A) = L$. Geben Sie A vollständig an.

Lösung:

Der nichtdeterministische endliche Automat A lautet:

$$A = (\{0,1\}, \{s_0, s_1, s_2\}, \delta, s_0, \{s_2\})$$

δ:



Aufgabe 7 ★

END-AH **Endliche Automaten**

Für einen Binärstring $w \in \{0,1\}^+$ bezeichne w_2 den Zahlenwert des Binärstrings als Dualzahl interpretiert.

a) Entwerfen Sie einen deterministischen endlichen Automaten A_1, der alle Binärstrings akzeptiert, die als Dualzahl interpretiert durch 2 teilbar sind. Der Automat soll also folgende Sprache L_1 erkennen:

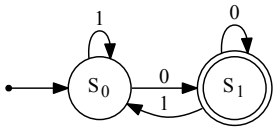
$$L_1 = \{w \in \{0,1\}^+ \mid w_2 \bmod 2 = 0\}$$

Lösung:

Der deterministische endliche Automat A_1 lautet

$$A_1 = (\{0,1\}, \{s_0, s_1\}, \delta, s_0, \{s_1\})$$

δ:



b) Entwerfen Sie einen deterministischen endlichen Automaten A_2, der alle Binärstrings akzeptiert, die als Dualzahl interpretiert durch 4 teilbar sind. Der Automat soll also folgende Sprache L_2 erkennen:

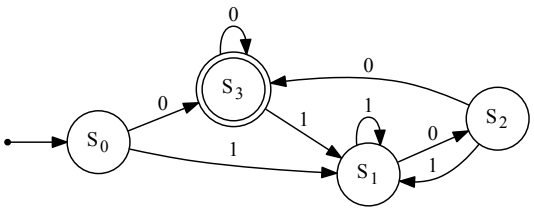
$$L_2 = \{w \in \{0,1\}^+ \mid w_2 \bmod 4 = 0\}$$

Lösung:

Der deterministische endliche Automat A_2 lautet

$$A_2 = (\{0,1\}, \{s_0, s_1, s_2, s_3\}, \delta, s_0, \{s_3\})$$

δ:



Aufgabe 8 ★★

END-AV **Endliche Automaten**

Ein Spiel sei definiert als zwei aufeinanderfolgende Würfe mit einem Würfel, der alle Zahlen zwischen 1 und 6 enthält. Das Spiel gilt als gewonnen, wenn die Summe der Augenzahlen durch 3 teilbar ist. Im Folgenden sei der Ablauf eines Spiels kodiert als ein Wort $w \in \{1, \ldots, 6\}^2$, wobei das erste Zeichen für den ersten Wurf und das zweite für den zweiten steht.

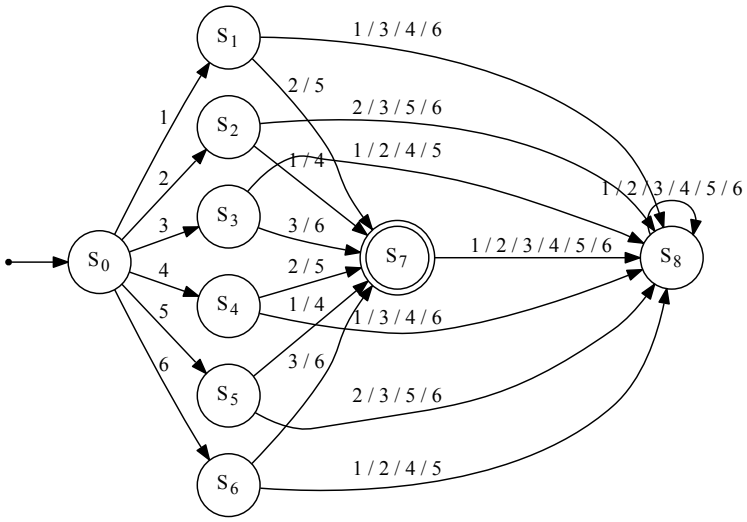
a) Entwickeln Sie einen deterministischen endlichen Automaten A, welcher die Kodierung eines Spielablaufs genau dann akzeptiert, wenn das Spiel als gewonnen gilt. Geben Sie den Automaten vollständig an.

Lösung:

Das Eingabealphabet E besteht aus den Zahlen 1 bis 6, die die möglichen Ergebnisse eines Wurfes darstellen. Der deterministische endliche Automat lässt sich demnach wie folgt darstellen:

$$A = (\{1,2,3,4,5,6\}, \{s_0, \ldots, s_8\}, \delta, s_0, \{s_7\})$$

δ:



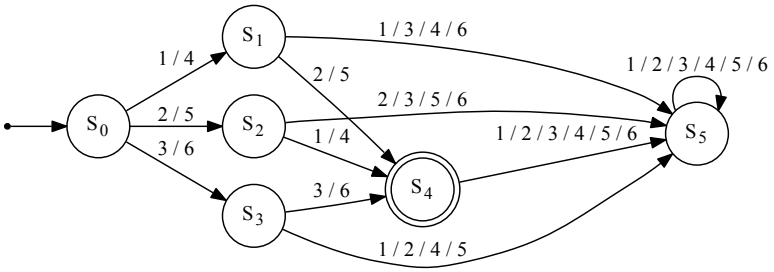
b) Minimieren Sie den Automaten A und geben Sie den minimierten Automaten A' vollständig an.

Lösung:

Beim Ausgangsautomaten A können immer zwei Zustände zusammengefasst werden, da diese, wie aus A ersichtlich, die gleichen Übergänge aufweisen. Auf eine Minimierung mithilfe des Algorithmus wird somit an dieser Stelle verzichtet (für Aufgaben zum Minimierungsalgorithmus, vgl. 3). Der minimierte Automat A' ergibt sich folglich zu:

$$A' = (\{1,2,3,4,5,6\}, \{s_0, \ldots, s_5\}, \delta', s_0, \{s_4\})$$

δ':



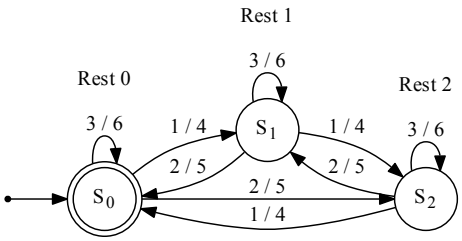
c) Verallgemeinern Sie den endlichen Automaten A bzw. A', sodass er bei einem beliebig langen Würfelspiel mit einem Würfel genau dann akzeptiert, wenn die Summe aller Würfe durch 3 teilbar ist. Geben Sie den verallgemeinerten Automaten A_{allg} ebenfalls vollständig an.

Lösung:

Der allgemeine Automat ist von der Anzahl der Zustände her gesehen einfacher als der Automat, der nur auf zwei Würfe beschränkt ist. Der Automat A bzw. A' muss im Gegenteil zu A_{allg} nämlich auch in der Nummer der Würfe unterscheiden. Bei A_{allg} ist es ausreichend, die möglichen Fehlmengen zur Quersumme (Rest) in den Zuständen zu kodieren.

$$A_{allg} = (\{1, 2, 3, 4, 5, 6\}, \{s_0, s_1, s_2\}, \delta, s_0, \{s_0\})$$

δ:



Aufgabe 9 ★

END-AF **Endliche Automaten und reguläre Ausdrücke**

a) Gegeben sei die Sprache L aller Wörter über $E = \{a, b, c\}$, die auf abc enden.

1) Geben Sie einen deterministischen endlichen Automaten $A_1 = (E, S, \delta, s_0, F)$ an mit $L(A_1) = L$. Geben Sie A_1 vollständig an.

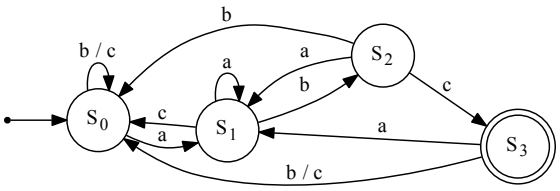
2) Geben Sie die Sprache L formal an.

Lösung:

Der deterministische endliche Automat A_1 ergibt sich zu:

$$A_1 = (\{a, b, c\}, \{s_0, s_1, s_2, s_3\}, \delta_1, s_0, \{s_3\})$$

δ_1:



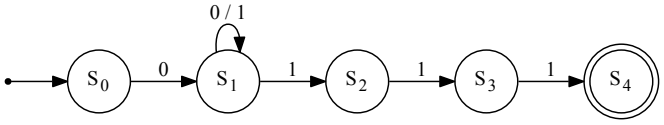
Die Sprache L lautet:

$$L(A_1) = \{w \in \{a, b, c\}^+ \mid \exists v \in \{a, b, c\}^\star : w = vabc\}$$

b) Gegeben sei der nichtdeterministische endliche Automat

$$A_2 = (\{0, 1\}, \{s_0, \ldots, s_4\}, \delta_2, s_0, \{s_4\})$$

δ_2:



1) Wandeln Sie den Automaten A_2 in einen deterministischen endlichen Automaten $A_2' = (E, S', F', \delta_2', s_0')$ um mit $L(A_2') = L(A_2)$. Geben Sie A_2' vollständig an.
2) Geben Sie einen regulären Ausdruck α an mit $L(\alpha) = L(A_2) = L(A_2')$.

Lösung:

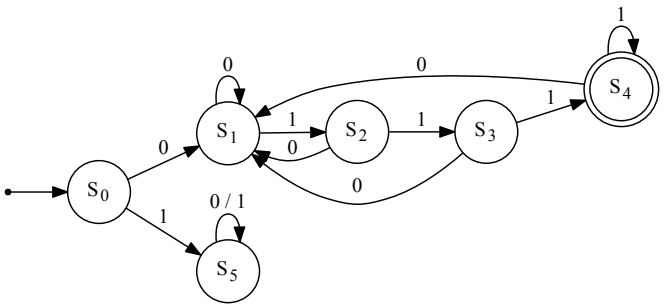
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_1\}$	$\emptyset$
$\{s_1\} \hat{=} s_1$	$\{s_1\}$	$\{s_1, s_2\}$
$\{s_1, s_2\} \hat{=} s_2$	$\{s_1\}$	$\{s_1, s_2, s_3\}$
$\{s_1, s_2, s_3\} \hat{=} s_3$	$\{s_1\}$	$\{s_1, s_2, s_3, s_4\}$
$\{s_1, s_2, s_3, s_4\} \hat{=} s_4$	$\{s_1\}$	$\{s_1, s_2, s_3, s_4\}$
$\emptyset \hat{=} s_5$	$\emptyset$	$\emptyset$

Daraus ergibt sich folgender deterministischer endlicher Automat A_2':

$$A_2' = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4, s_5\}, \delta_2', s_0, \{s_4\})$$

δ_2':



Anmerkung:

Der deterministische endliche Automaten hätte auch ohne Potenzmengenkonstruktion durch folgende informale Überlegung erstellen können: Zunächst wird ein Pfad vom Anfangszustand zum Endzustand über $(0 \to 1 \to 1 \to 1)$ hergestellt, der den Mittelteil mit beliebig vielen Nullen und Einsen weglässt. Die noch nicht definierten Übergänge können durch folgende Überlegungen ergänzt werden:

- Da zu Beginn immer eine 0 stehen muss, muss A_2' von s_0 über 1 in eine Senke führen.
- Da am Ende genau drei 0 kommen müssen, müssen alle übrigen Zuständen über 0 nach s_1 wechseln.

Der Regulärer Ausdruck kann direkt aus A_2 abgelesen werden und lautet:

$$\alpha = 0(0 + 1)^\star 111$$

Aufgabe 10 ⋆

END-AG **Endliche Automaten und reguläre Ausdrücke**

Gegeben sei eine Sprache L mit

$$L = \{w \in \{0,1\}^\star \mid \exists v \in \{0,1\}^\star : w = 01v01\}.$$

Die Sprache L enthält somit alle Wörter, die sowohl mit 01 beginnen als auch mit 01 enden.

a) Erstellen Sie eine rechtslineare Grammatik, welche die Sprache L erzeugt. Geben Sie die Grammatik vollständig an und leiten Sie das Testwort 010010001 ab.

Lösung:

Die rechtslineare Grammatik mit $P \subseteq N \times (T \cup TN \cup \{\lambda\})$ ergibt sich zu:

$$\begin{aligned} G &= (N, T, P, S) \\ N &= \{A, B, C, S\} \\ T &= \{0, 1\} \\ P &= \{S \to 0A, \\ &\quad\ A \to 1B, \\ &\quad\ B \to 0B \mid 1B \mid 0C, \\ &\quad\ C \to 1\} \end{aligned}$$

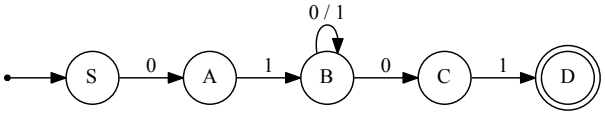
Produktion des Testwortes:

$$\begin{aligned} S &\Rightarrow 0A \Rightarrow 01B \Rightarrow 010B \Rightarrow 0100B \Rightarrow 01001B \Rightarrow 010010B \\ &\Rightarrow 0100100B \Rightarrow 01001000C \Rightarrow 010010001 \end{aligned}$$

b) Gegeben sei ein nichtdeterministischer endlicher Automat A, der die Sprache L erkennt, mit

$$A = (\{0,1\}, \{S, A, B, C, D\}, \delta, S, \{D\})$$

δ:



Geben Sie einen deterministischen endlichen Automaten $A' = (\{0,1\}, S', \delta', s_0, F')$ an mit $L(A') = L(A) = L$. (Beachten Sie, dass die Zustände des Automaten A zur Verdeutlichung der Ähnlichkeit zur Grammatik aus Aufgabenteil a) mit Großbuchstaben bezeichnet sind; insbesondere ist dadurch S ein Zustand, aber S' eine Zustandsmenge.)

Lösung:

Anmerkung:

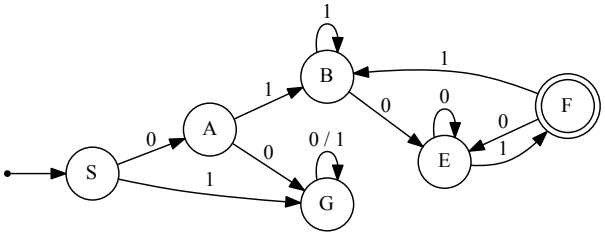
Beachten Sie die Parallelität in den Strukturen der Grammatik und des nichtdeterministischen Automaten.

Durch Potenzmengenkonstruktion ergibt sich:

	0	1
$\{S\} \hat{=} S$	$\{A\}$	$\emptyset$
$\{A\} \hat{=} A$	$\emptyset$	$\{B\}$
$\{B\} \hat{=} B$	$\{B, C\}$	$\{B\}$
$\{B, C\} \hat{=} E$	$\{B, C\}$	$\{B, D\}$
$\{B, D\} \hat{=} F$	$\{B, C\}$	$\{B\}$
$\emptyset \hat{=} G$	$\emptyset$	$\emptyset$

$$A' = (\{0,1\}, \{S, A, B, E, F, G\}, \delta', S, \{F\})$$

δ':



c) Geben Sie einen regulären Ausdruck α an mit $L(\alpha) = L(A) = L(A') = L$.

Lösung:

$\alpha = 01(0 + 1)^\star 01$

Aufgabe 11 ⋆

END-AC **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei der reguläre Ausdruck α mit

$$\alpha = 01(0 + 1)^\star 10^\star$$

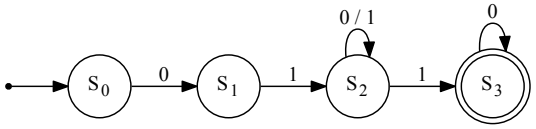
Entwickeln Sie einen nichtdeterministischen endlichen Automaten A und formen Sie diesen anschließend in einen deterministischen endlichen Automaten A' um mit $L(\alpha) = L(A) = L(A')$. Geben Sie die Automaten A und A' vollständig an.

Lösung:

Der nichtdeterministische endliche Automat A ergibt sich zu:

$$A = (\{0, 1\}, \{s_0, s_1, s_2, s_3\}, \delta, s_0, \{s_3\})$$

δ:



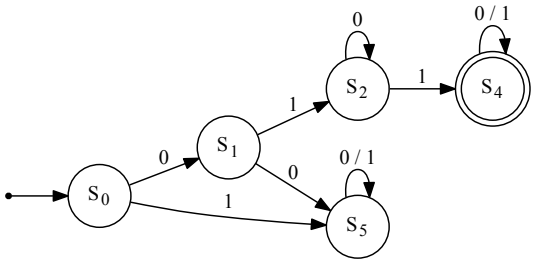
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_1\}$	$\emptyset$
$\{s_1\} \hat{=} s_1$	$\emptyset$	$\{s_2\}$
$\{s_2\} \hat{=} s_2$	$\{s_2\}$	$\{s_2, s_3\}$
$\{s_2, s_3\} \hat{=} s_4$	$\{s_2, s_3\}$	$\{s_2, s_3\}$
$\emptyset \hat{=} s_5$	$\emptyset$	$\emptyset$

Der deterministische endliche Automat A' ergibt sich zu:

$$A' = (\{0, 1\}, \{s_0, s_1, s_2, s_4, s_5\}, \delta', s_0, \{s_4\})$$

δ':



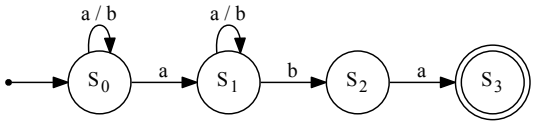
Aufgabe 12 ★

END-AL **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei folgender nichtdeterministischer endlicher Automat A

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

δ:



a) Wandeln Sie A in einen äquivalenten deterministischen endlichen Automaten A' um mit $L(A') = L(A)$ und geben Sie diesen vollständig an.

Lösung:

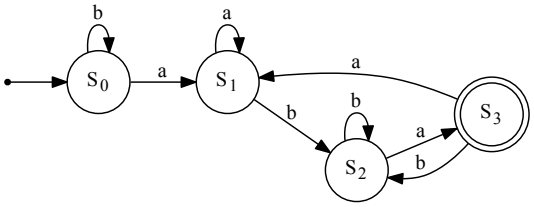
Umwandlung durch Potenzmengenkonstruktion:

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

Der deterministische endliche Automat A' ergibt sich zu:

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

δ:



b) Geben Sie die Sprache, die von dem Automaten erkannt wird, als regulären Ausdruck α an, sodass $L(\alpha) = L(A)$ gilt.

Lösung:

Aus dem nichtdeterministischen endlichen Automaten ergibt sich folgender regulärer Ausdruck:

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

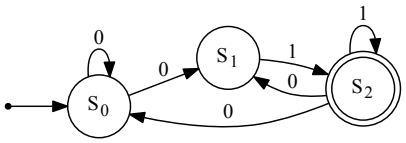
Aufgabe 13 ⋆

END-AN **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei folgender nichtdeterministischer endlicher Automat A

$$A = (\{0, 1\}, \{s_0, s_1, s_2\}, \delta, s_0, \{s_2\})$$

δ:



a) Konstruieren Sie zu dem nichtdeterministischen endlichen Automaten A einen äquivalenten deterministischen endlichen Automaten A' mit $L(A') = L(A)$ und geben Sie diesen vollständig an.

Lösung:

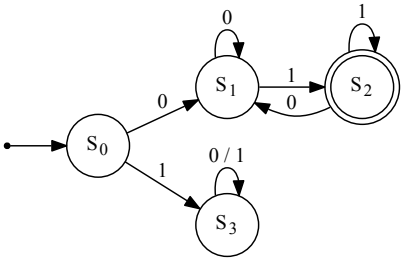
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_0, s_1\}$	$\emptyset$
$\{s_0, s_1\} \hat{=} s_1$	$\{s_0, s_1\}$	$\{s_2\}$
$\{s_2\} \hat{=} s_2$	$\{s_0, s_1\}$	$\{s_2\}$
$\emptyset \hat{=} s_3$	$\emptyset$	$\emptyset$

Es ergibt sich folgender Automat:

$$A' = (\{0, 1\}, \{s_0, s_1, s_2, s_3\}, \delta', a_0, \{s_2\})$$

δ':



b) Geben Sie einen regulären Ausdruck α an mit $L(\alpha) = L(A) = L(A')$.

Lösung:

Der entsprechende reguläre Ausdruck α lautet (dabei sei zusätzlich zu den Operatoren der regulären Ausdrücke noch der Operator "+" zugelassen mit $x^+ =_{def} xx^\star$):

$$\begin{aligned}\alpha &= 0^\star 01(1 + 00^\star 01 + 01)^\star \text{ (aus } A \text{ abgelesen)}\\ &\widehat{=} 0^+1(1 + 0^+01 + 01)^\star \text{ oder}\\ \alpha &= 0(0 + 1)^\star 1 \text{ (aus } A' \text{ abgelesen)}\end{aligned}$$

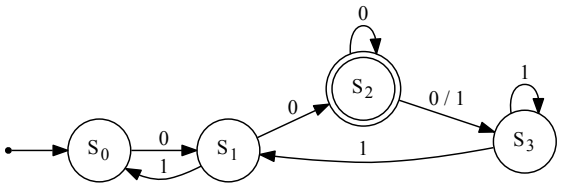
Aufgabe 14 ⋆⋆

END-AI **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei der nichtdeterministische endliche Automat A

$$A = (\{0, 1\}, \{s_0, s_1, s_2, s_3\}, \delta, s_0, \{s_2\})$$

δ:



Konstruieren Sie zu A einen äquivalenten deterministischen endlichen Automaten A'. Geben Sie den Automaten A' vollständig an.

Lösung:

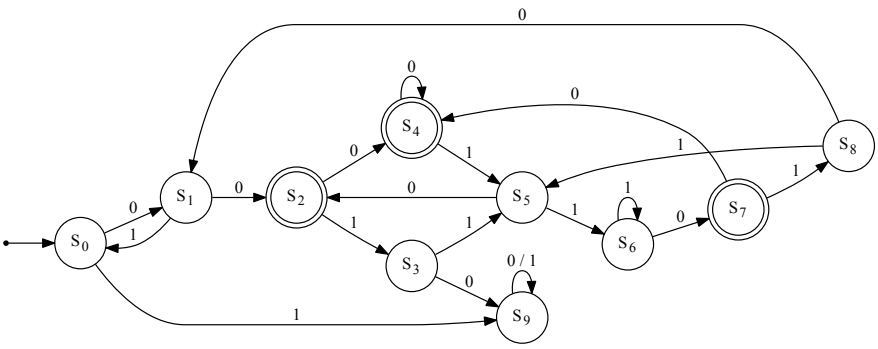
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\}\widehat{=}s_0$	$\{s_1\}$	$\emptyset$
$\{s_1\}\widehat{=}s_1$	$\{s_2\}$	$\{s_0\}$
$\{s_2\}\widehat{=}s_2$	$\{s_2, s_3\}$	$\{s_3\}$
$\{s_3\}\widehat{=}s_3$	$\emptyset$	$\{s_1, s_3\}$
$\{s_2, s_3\}\widehat{=}s_4$	$\{s_2, s_3\}$	$\{s_1, s_3\}$
$\{s_1, s_3\}\widehat{=}s_5$	$\{s_2\}$	$\{s_0, s_1, s_3\}$
$\{s_0, s_1, s_3\}\widehat{=}s_6$	$\{s_1, s_2\}$	$\{s_0, s_1, s_3\}$
$\{s_1, s_2\}\widehat{=}s_7$	$\{s_2, s_3\}$	$\{s_0, s_3\}$
$\{s_0, s_3\}\widehat{=}s_8$	$\{s_1\}$	$\{s_1, s_3\}$
$\emptyset\widehat{=}s_9$	$\emptyset$	$\emptyset$

Es ergibt sich der folgende Automat:

$$A' = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9\}, \delta', s_0, \{s_2, s_4, s_7\})$$

δ':



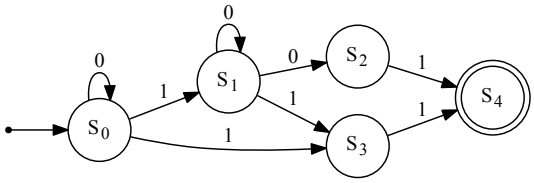
Aufgabe 15 ★⋆

END-AM **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Wandeln Sie den folgenden nichtdeterministischen endlichen Automaten A in einen äquivalenten deterministischen endlichen Automaten A' um mit $L(A') = L(A)$. Geben Sie A' vollständig an.

$$A = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta, s_0, \{s_4\})$$

δ:



Lösung:

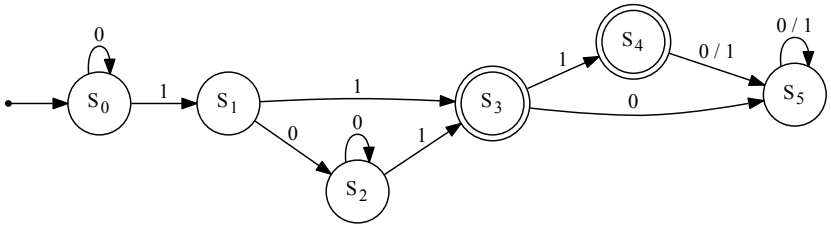
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_0\}$	$\{s_1, s_3\}$
$\{s_1, s_3\} \hat{=} s_1$	$\{s_1, s_2\}$	$\{s_3, s_4\}$
$\{s_1, s_2\} \hat{=} s_2$	$\{s_1, s_2\}$	$\{s_3, s_4\}$
$\{s_3, s_4\} \hat{=} s_3$	$\emptyset$	$\{s_4\}$
$\{s_4\} \hat{=} s_4$	$\emptyset$	$\emptyset$
$\emptyset \hat{=} s_5$	$\emptyset$	$\emptyset$

Der deterministische endliche Automat A' lautet:

$$A' = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4, s_5\}, \delta', s_0, \{s_3, s_4\})$$

δ':



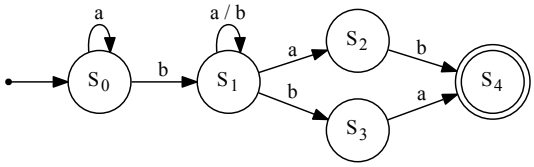
Aufgabe 16 ★★

END-AX **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei der nichtdeterministische endliche Automat

$$A = (\{a, b\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta, s_0, \{s_4\})$$

δ:



a) Konstruieren Sie einen äquivalenten deterministischen endlichen Automaten A', sodass gilt $L(A') = L(A)$. Geben Sie A' vollständig an.

Lösung:

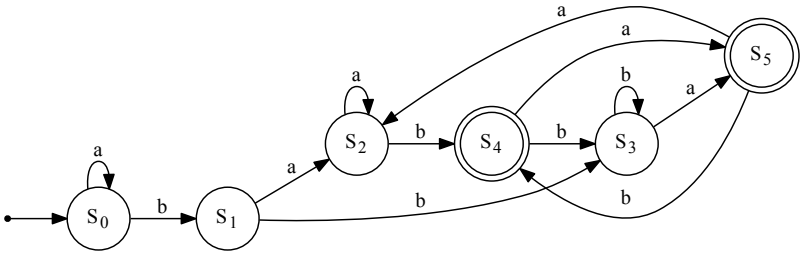
Umwandlung durch Potenzmengenkonstruktion:

	a	b
$\{s_0\} \hat{=} s_0$	$\{s_0\}$	$\{s_1\}$
$\{s_1\} \hat{=} s_1$	$\{s_1, s_2\}$	$\{s_1, s_3\}$
$\{s_1, s_2\} \hat{=} s_2$	$\{s_1, s_2\}$	$\{s_1, s_3, s_4\}$
$\{s_1, s_3\} \hat{=} s_3$	$\{s_1, s_2, s_4\}$	$\{s_1, s_3\}$
$\{s_1, s_3, s_4\} \hat{=} s_4$	$\{s_1, s_2, s_4\}$	$\{s_1, s_3\}$
$\{s_1, s_2, s_4\} \hat{=} s_5$	$\{s_1, s_2\}$	$\{s_1, s_3, s_4\}$

Daraus ergibt sich folgender deterministischer endlicher Automat:

$$A' = (\{a, b\}, \{s_0, s_1, s_2, s_3, s_4, s_5\}, \delta', s_0, \{s_4, s_5\})$$

δ':



b) Geben Sie die Sprache, die von dem Automaten erkannt wird, als regulären Ausdruck α an mit $L(\alpha) = L(A)$.

Lösung:

$\alpha = a^\star b(a + b)^\star(ab + ba)$

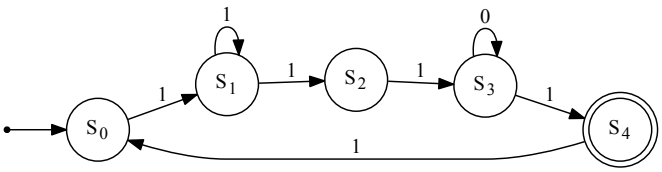
Aufgabe 17 ★★⋆

END-AW **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei folgender nichtdeterministischer endlicher Automat A. Konstruieren Sie einen äquivalenten deterministischen endlichen Automaten A', sodass gilt $L(A') = L(A)$. Geben Sie A' vollständig an.

$$A = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta, s_0, \{s_4\})$$

δ:



Lösung:

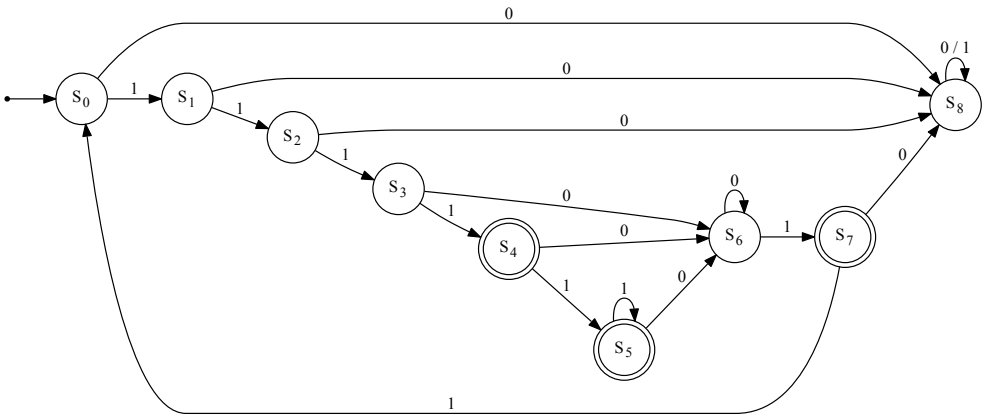
Umwandlung durch Potenzmengenkonstruktion:

Zustandsmenge	0	1
$\{s_0\} \hat{=} s_0$	$\emptyset$	$\{s_1\}$
$\{s_1\} \hat{=} s_1$	$\emptyset$	$\{s_1, s_2\}$
$\{s_1, s_2\} \hat{=} s_2$	$\emptyset$	$\{s_1, s_2, s_3\}$
$\{s_1, s_2, s_3\} \hat{=} s_3$	$\{s_3\}$	$\{s_1, s_2, s_3, s_4\}$
$\{s_1, s_2, s_3, s_4\} \hat{=} s_4$	$\{s_3\}$	$\{s_0, s_1, s_2, s_3, s_4\}$
$\{s_0, s_1, s_2, s_3, s_4\} \hat{=} s_5$	$\{s_3\}$	$\{s_0, s_1, s_2, s_3, s_4\}$
$\{s_3\} \hat{=} s_6$	$\{s_3\}$	$\{s_4\}$
$\{s_4\} \hat{=} s_7$	$\emptyset$	$\{s_0\}$
$\emptyset \hat{=} s_8$	$\emptyset$	$\emptyset$

Somit ergibt sich folgender endlicher Automat:

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

δ':



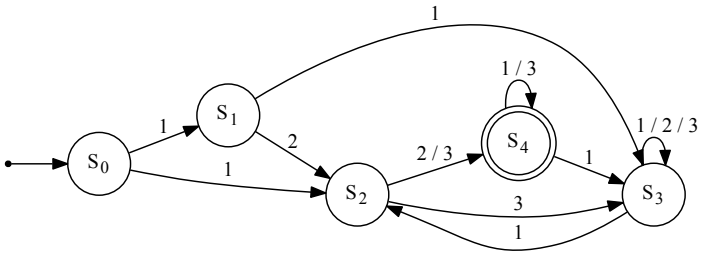
Aufgabe 18 ★★⋆

END-AY **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Konstruieren Sie zu dem nichtdeterministischen endlichen Automaten A einen äquivalenten deterministischen endlichen Automaten A' mit $L(A') = L(A)$. Geben Sie den Automaten A' vollständig an.

$$A = (\{1, 2, 3\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta_0, s_0, \{s_4\})$$

δ_0:



Lösung:

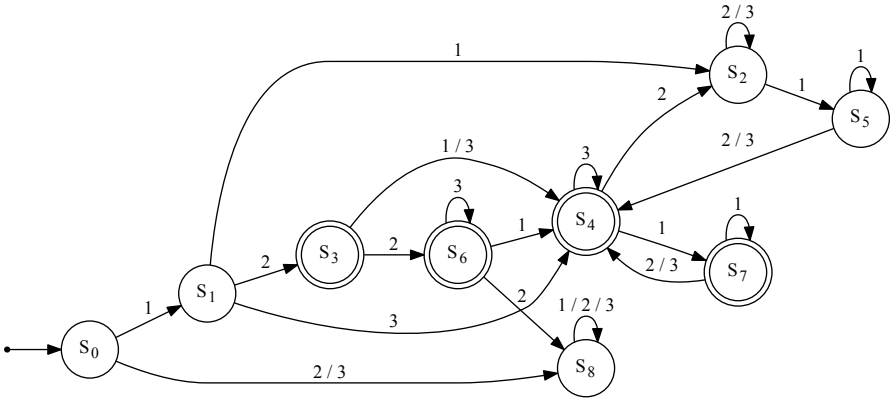
Umwandlung durch Potenzmengenkonstruktion:

	1	2	3
$\{s_0\} \hat{=} s_0$	$\{s_1, s_2\}$	$\emptyset$	$\emptyset$
$\{s_1, s_2\} \hat{=} s_1$	$\{s_3\}$	$\{s_2, s_4\}$	$\{s_3, s_4\}$
$\{s_3\} \hat{=} s_2$	$\{s_2, s_3\}$	$\{s_3\}$	$\{s_3\}$
$\{s_2, s_4\} \hat{=} s_3$	$\{s_3, s_4\}$	$\{s_4\}$	$\{s_3, s_4\}$
$\{s_3, s_4\} \hat{=} s_4$	$\{s_2, s_3, s_4\}$	$\{s_3\}$	$\{s_3, s_4\}$
$\{s_2, s_3\} \hat{=} s_5$	$\{s_2, s_3\}$	$\{s_3, s_4\}$	$\{s_3, s_4\}$
$\{s_4\} \hat{=} s_6$	$\{s_3, s_4\}$	$\emptyset$	$\{s_4\}$
$\{s_2, s_3, s_4\} \hat{=} s_7$	$\{s_2, s_3, s_4\}$	$\{s_3, s_4\}$	$\{s_3, s_4\}$
$\emptyset \hat{=} s_8$	$\emptyset$	$\emptyset$	$\emptyset$

Somit ergibt sich folgender endlicher Automat:

$$A' = (\{1, 2, 3\}, \{s_0, \ldots, s_8\}, \delta_1, s_0, \{s_3, s_4, s_6, s_7\})$$

δ_1:



Aufgabe 19 ★⋆

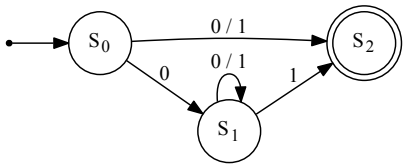
END-AK **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Konstruieren Sie zu den nachfolgenden nichtdeterministischen endlichen Automaten $A_i, i \in \{1, 2\}$ äquivalente deterministische endliche Automaten A_i' mit $L(A_i) = L(A_i')$ und geben Sie diese vollständig an. Beschreiben Sie zudem die Sprachen, welche von den Automaten erkannt werden, als reguläre Ausdrücke α_i, sodass $L(\alpha_i) = L(A_i)$ gilt.

a) A_1 sei gegeben durch

$$A_1 = (\{0, 1\}, \{s_0, s_1, s_2\}, \delta_1, s_0, \{s_2\})$$

δ_1:



Lösung:

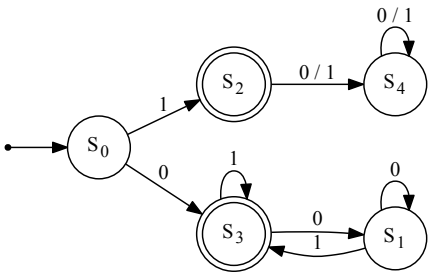
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_1, s_2\}$	$\{s_2\}$
$\{s_1, s_2\} \hat{=} s_3$	$\{s_1\}$	$\{s_1, s_2\}$
$\{s_2\} \hat{=} s_2$	$\emptyset$	$\emptyset$
$\{s_1\} \hat{=} s_1$	$\{s_1\}$	$\{s_1, s_2\}$
$\emptyset \hat{=} s_4$	$\emptyset$	$\emptyset$

Daraus ergibt sich folgender deterministischer endlicher Automat:

$$A_1' = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta_1', s_0, \{s_2, s_3\})$$

δ_1':



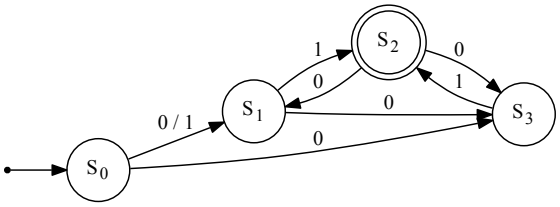
mit dem regulären Ausdruck:

$$\alpha_1 = 0 + 1 + 0(0 + 1)^\star 1 \text{ oder}$$
$$\alpha_1 = 1 + 0((0 + 1)^\star 1)^\star$$

b) A_2 sei gegeben durch

$$A_2 = (\{0, 1\}, \{s_0, s_1, s_2, s_3, \}, \delta_2, s_0, \{s_2\})$$

δ_2:



Lösung:

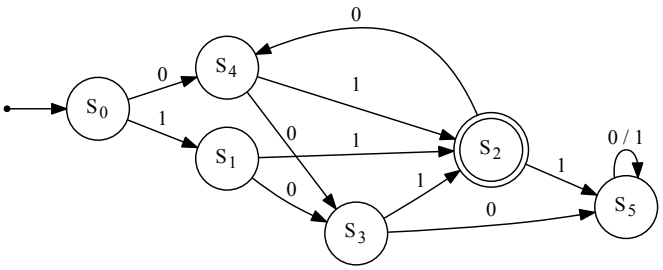
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_1, s_3\}$	$\{s_1\}$
$\{s_1, s_3\} \hat{=} s_4$	$\{s_3\}$	$\{s_2\}$
$\{s_1\} \hat{=} s_1$	$\{s_3\}$	$\{s_2\}$
$\{s_2\} \hat{=} s_2$	$\{s_1, s_3\}$	$\emptyset$
$\{s_3\} \hat{=} s_3$	$\emptyset$	$\{s_2\}$
$\emptyset \hat{=} s_5$	$\emptyset$	$\emptyset$

Daraus ergibt sich folgender deterministischer endlicher Automat:

$$A_2' = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4, s_5\}, \delta_2', s_0, \{s_2\})$$

δ_2':



mit dem regulären Ausdruck:

$$\alpha_2 = ((0+1)(01+1)+01)(01+001)^\star \text{ oder}$$
$$\alpha_2 = (0(01+1)+1(01+1))(0(1+01))^\star$$

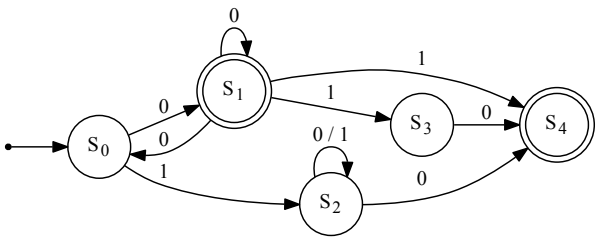
Aufgabe 20 $\star^{\star\star}$

END-AZ **Umwandlung nichtdeterministischer in deterministische endliche Automaten**

Gegeben sei folgender nichtdeterministischer endlicher Automat A

$$A = (E, S, \delta, s_0, F) = (\{0,1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta, s_0, \{s_1, s_4\})$$

δ:



a) Geben Sie die Zustandsübergangstabelle für die Potenzmengen-Konstruktion eines zu A äquivalenten deterministischen endlichen Automaten A' an. Geben Sie A' vollständig an.

Lösung:

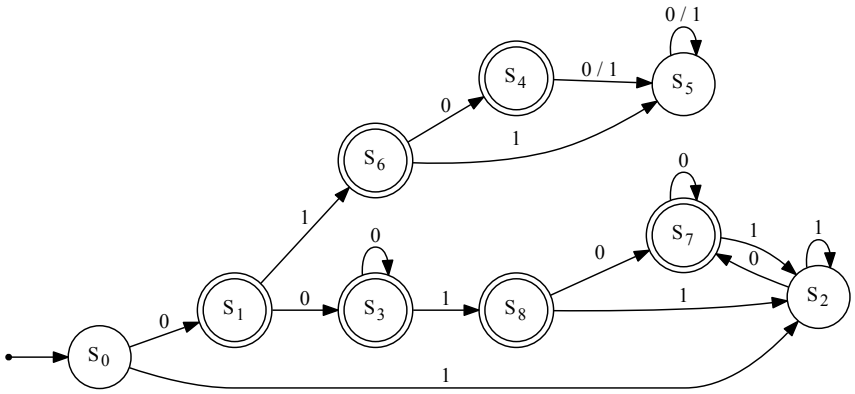
Umwandlung durch Potenzmengenkonstruktion:

	0	1
$\{s_0\} \hat{=} s_0$	$\{s_1\}$	$\{s_2\}$
$\{s_1\} \hat{=} s_1$	$\{s_0, s_1\}$	$\{s_3, s_4\}$
$\{s_2\} \hat{=} s_2$	$\{s_2, s_4\}$	$\{s_2\}$
$\{s_0, s_1\} \hat{=} s_3$	$\{s_0, s_1\}$	$\{s_2, s_3, s_4\}$
$\{s_3, s_4\} \hat{=} s_6$	$\{s_4\}$	$\emptyset$
$\{s_2, s_4\} \hat{=} s_7$	$\{s_2, s_4\}$	$\{s_2\}$
$\{s_2, s_3, s_4\} \hat{=} s_8$	$\{s_2, s_4\}$	$\{s_2\}$
$\{s_4\} \hat{=} s_4$	$\emptyset$	$\emptyset$
$\emptyset \hat{=} s_5$	$\emptyset$	$\emptyset$

Somit ergibt sich folgender deterministischer endlicher Automat:

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

δ':



b) Geben Sie die Sprache L als regulären Ausdruck α an mit $L(\alpha) = L(A') = L(A)$.

Lösung:

$\alpha = (000^\star + \emptyset^\star)1(0 + 1)^\star 0 + 00^\star(1 + 10 + \emptyset^\star)$

3 Minimierung endlicher Automaten

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=348

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

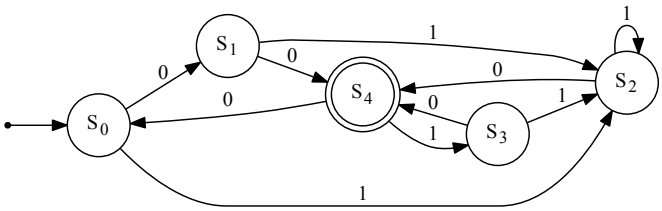
Aufgabe 21 ★

MIN-AA **Minimierung endlicher Automaten**

Minimieren Sie den endlichen Automaten A und geben Sie den minimierten Automaten vollständig an.

$$A = (\{0, 1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta, s_0, \{s_4\})$$

δ:



Lösung:

Zustandsübergangstabelle:

δ	0	1
s_0	s_1	s_2
s_1	s_4	s_2
s_2	s_4	s_2
s_3	s_4	s_2
s_4	s_0	s_3

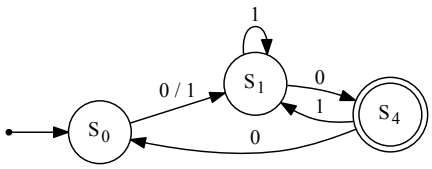
Minimierungstabelle ($\times_k$ in einer Zelle bedeutet, dass die entsprechenden Zustände nicht k-äquivalent sind, aber, falls $k > 0$, $(k-1)$-äquivalent):

s_1	$\times_1$			
s_2	$\times_1$	$-$		
s_3	$\times_1$	$-$	$-$	
s_4	$\times_0$	$\times_0$	$\times_0$	$\times_0$
	s_0	s_1	s_2	s_3

$\Rightarrow$ s_1, s_2, s_3 sind äquivalent ($s_1 \sim s_2 \sim s_3$) und werden im neuen Automaten A' zu s_1 zusammengefasst.

$$A' = (\{0, 1\}, \{s_0, s_1, s_4\}, \delta', s_0, \{s_4\})$$

δ':



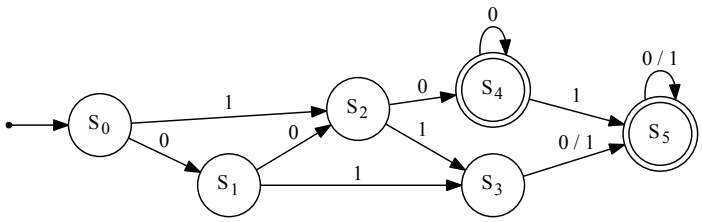
Aufgabe 22 ★

MIN-AB **Minimierung endlicher Automaten**

Gegeben sei der endliche Automat A. Zählen Sie für $k = 0, 1, 2, \ldots$ die Mengen aller zueinander k-äquivalenten Zustände auf. Lassen sich mehrere Zustände zu einem zusammenfassen?

$$A = (\{0, 1\}, \{s_0, \ldots, s_5\}, \delta, s_0, \{s_4, s_5\})$$

δ:



Lösung:

Zustandsübergangstabelle:

δ	0	1
s_0	s_1	s_2
s_1	s_2	s_3
s_2	s_4	s_3
s_3	s_5	s_5
s_4	s_4	s_5
s_5	s_5	s_5

Minimierungstabelle:

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

- Mengen 0-äquivalenter Zustände: $\{s_0, s_1, s_2, s_3\}; \{s_4, s_5\}$
- Mengen 1-äquivalenter Zustände: $\{s_0, s_1\}; \{s_2\}; \{s_3\}; \{s_4, s_5\}$
- Mengen 2-äquivalenter Zustände: $\{s_0\}; \{s_1\}; \{s_2\}; \{s_3\}; \{s_4, s_5\}$
- Mengen 3- und mehr-äquivalenter Zustände: $\{s_0\}; \{s_1\}; \{s_2\}; \{s_3\}; \{s_4, s_5\}$

$\Rightarrow$ Mengen äquivalenter Zustände: $\{s_0\}; \{s_1\}; \{s_2\}; \{s_3\}; \{s_4, s_5\}$

Die Zustände s_4 und s_5 lassen sich zu einem neuen Endzustand zusammenfassen, da sie für alle k zueinander k-äquivalent sind.

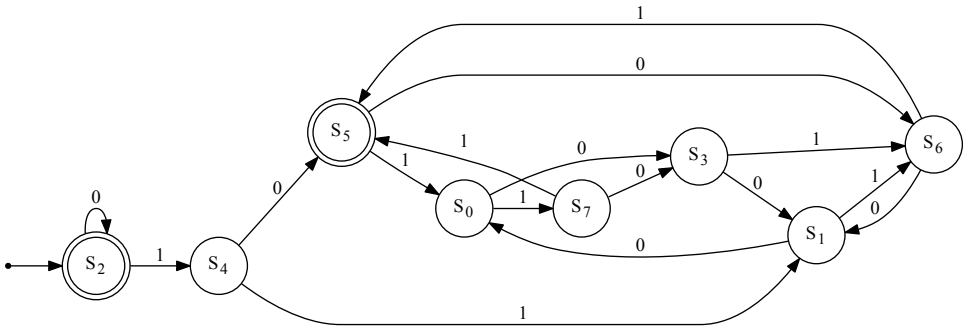
Aufgabe 23 ★

MIN-AE **Minimierung endlicher Automaten**

Gegeben sei der deterministische endliche Automat $A = (E, S, \delta, s_0, F)$ mit

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

δ:



Minimieren Sie A und geben Sie den minimierten Automaten $A' = (E, S', \delta', s_0', F')$ vollständig an.

Lösung:

Zustandsüberführungstabelle:

δ	0	1
s_0	s_3	s_7
s_1	s_0	s_6
s_2	s_2	s_4
s_3	s_1	s_6
s_4	s_5	s_1
s_5	s_6	s_0
s_6	s_1	s_5
s_7	s_3	s_5

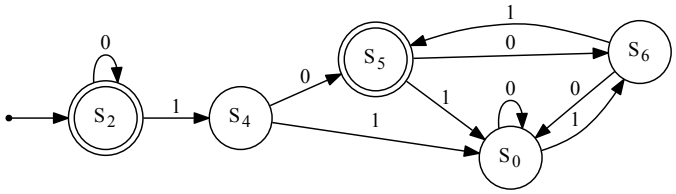
Minimierungstabelle:

s_1	–						
s_2	$\times_0$	$\times_0$					
s_3	–	–	$\times_0$				
s_4	$\times_1$	$\times_1$	$\times_0$	$\times_1$			
s_5	$\times_0$	$\times_0$	$\times_1$	$\times_0$	$\times_0$		
s_6	$\times_1$	$\times_1$	$\times_0$	$\times_1$	$\times_1$	$\times_0$	
s_7	$\times_1$	$\times_1$	$\times_0$	$\times_1$	$\times_1$	$\times_0$	–
	s_0	s_1	s_2	s_3	s_4	s_5	s_6

Es gilt $s_0 \sim s_1 \sim s_3$ und $s_6 \sim s_7$. Mit $s_0 \widehat{=} \{s_0, s_1, s_3\}$, $s_6 \widehat{=} \{s_6, s_7\}$ ergibt sich:

$$A' = (E, S', \delta', s'_0, F') = (\{0,1\}, \{s_0, s_2, s_4, s_5, s_6\}, \delta', s_2, \{s_2, s_5\})$$

δ':



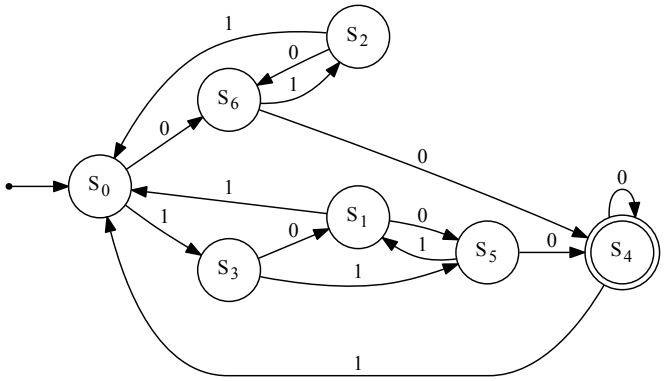
Aufgabe 24 ★⋆

MIN-AD **Minimierung endlicher Automaten**

Gegeben sei der deterministische endliche Automat $A = (E, S, \delta, s_0, F)$. Durch das abgebildete Zustandsdiagramm sei δ definiert.

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

δ:



Minimieren Sie A und geben Sie den minimierten Automaten $A' = (E, S', \delta', s'_0, F')$ vollständig an.

Lösung:

Zustandsübergangstabelle:

δ	0	1
s_0	s_6	s_3
s_1	s_5	s_0
s_2	s_6	s_0
s_3	s_1	s_5
s_4	s_4	s_0
s_5	s_4	s_1
s_6	s_4	s_2

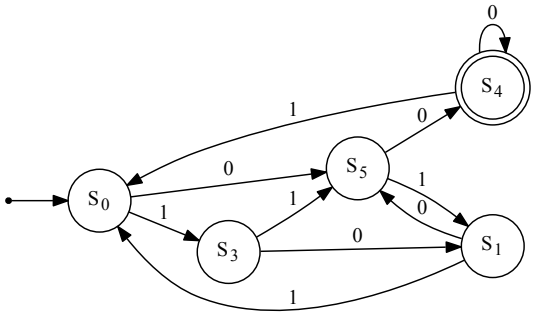
Minimierungstabelle:

s_1	$\times_3$					
s_2	$\times_3$	$-$				
s_3	$\times_2$	$\times_2$	$\times_2$			
s_4	$\times_0$	$\times_0$	$\times_0$	$\times_0$		
s_5	$\times_1$	$\times_1$	$\times_1$	$\times_1$	$\times_0$	
s_6	$\times_1$	$\times_1$	$\times_1$	$\times_1$	$\times_0$	$-$
	s_0	s_1	s_2	s_3	s_4	s_5

Es gilt $s_1 \sim s_2$ und $s_5 \sim s_6$. Mit $s_1 \widehat{=} \{s_1, s_2\}$, $s_5 \widehat{=} \{s_5, s_6\}$ im neuen Automaten A' ergibt sich:

$$A' = (E, S', \delta', s_0', F') = (\{0, 1\}, \{s_0, s_1, s_3, s_4, s_5\}, \delta', s_0, \{s_4\})$$

δ':



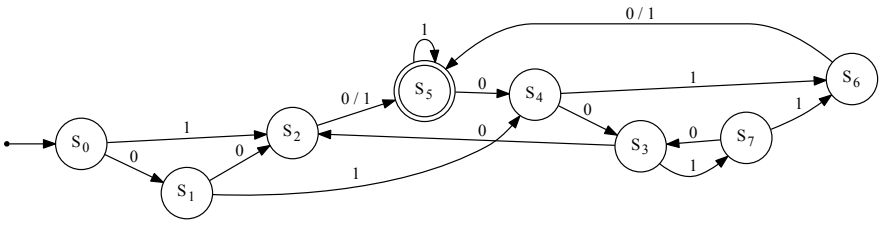
Aufgabe 25 ★⋆

MIN-AF **Minimierung endlicher Automaten**

Gegeben sei der deterministische endliche Automat $A = (E, S, \delta, s_0, F)$. Durch das abgebildete Zustandsdiagramm sei δ definiert.

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

δ:



Minimieren Sie A und geben Sie den minimierten Automaten vollständig an.

Lösung:

Zustandsüberführungstabelle:

δ	0	1
s_0	s_1	s_2
s_1	s_2	s_4
s_2	s_5	s_5
s_3	s_2	s_7
s_4	s_3	s_6
s_5	s_4	s_5
s_6	s_5	s_5
s_7	s_3	s_6

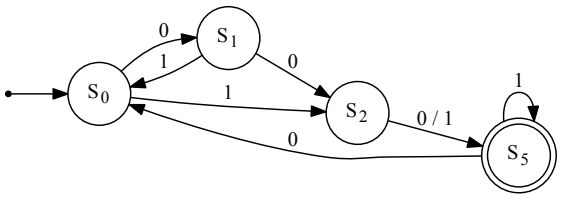
Minimierungstabelle:

s_1	$\times_2$						
s_2	$\times_1$	$\times_1$					
s_3	$\times_2$	$-$	$\times_1$				
s_4	$-$	$\times_2$	$\times_1$	$\times_2$			
s_5	$\times_0$	$\times_0$	$\times_0$	$\times_0$	$\times_0$		
s_6	$\times_1$	$\times_1$	$-$	$\times_1$	$\times_1$	$\times_0$	
s_7	$-$	$\times_2$	$\times_1$	$\times_2$	$-$	$\times_0$	$\times_1$
	s_0	s_1	s_2	s_3	s_4	s_5	s_6

Es gilt $s_0 \sim s_4 \sim s_7$, $s_1 \sim s_3$ und $s_2 \sim s_6$. Mit $s_0 \widehat{=} \{s_0, s_4, s_7\}$, $s_1 \widehat{=} \{s_1, s_3\}$, $s_2 \widehat{=} \{s_2, s_6\}$ ergibt sich:

$$A' = (E, S', \delta', s_0, F') = (\{0, 1\}, \{s_0, s_1, s_2, s_5\}, \delta', s_0, \{s_5\})$$

δ':



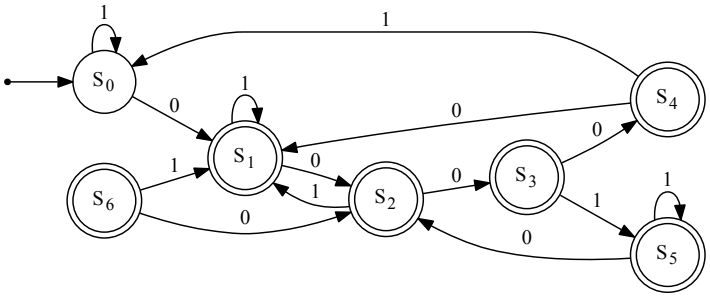
Aufgabe 26 ★⋆

MIN-AG **Minimierung endlicher Automaten**

Gegeben sei der endliche Automat

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

δ:



a) Minimieren Sie den endlichen Automaten A.

Lösung:

Zustand s_6 wird nie erreicht und braucht deshalb nicht betrachtet zu werden. Wir betrachten im Folgenden nur den vereinfachten Automaten.

Zustandsübergangstabelle:

δ	0	1
s_0	s_1	s_0
s_1	s_2	s_1
s_2	s_3	s_1
s_3	s_4	s_5
s_4	s_1	s_0
s_5	s_2	s_5

Minimierungstabelle:

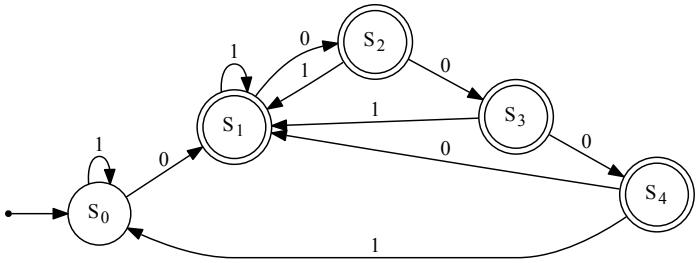
s_1	$\times_0$				
s_2	$\times_0$	$\times_3$			
s_3	$\times_0$	$\times_2$	$\times_2$		
s_4	$\times_0$	$\times_1$	$\times_1$	$\times_1$	
s_5	$\times_0$	$-$	$\times_3$	$\times_2$	$\times_1$
	s_0	s_1	s_2	s_3	s_4

Es gilt $s_1 \sim s_5$.

Der minimale Automat ist mit $s_1 \widehat{\equiv} \{s_1, s_5\}$:

$$A' = (\{0,1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta', s_0, \{s_1, s_2, s_3, s_4\})$$

δ':



b) Würde sich die Anzahl an Zuständen des minimierten Automaten im Vergleich zu a) ändern, falls ein anderer Knoten Startknoten wäre als s_0?

Lösung:

Falls die Knoten s_1, s_2, s_3, s_4 oder s_5 Startknoten wären, würde sich zwar der Automat verändern, nicht aber die Anzahl an Zuständen des minimalen Automaten (da man von jedem Knoten aus zu s_0 kommt). Allein Knoten s_6 bildet die Ausnahme: wäre er Startknoten, so müsste er bei Anwendung des Algorithmus berücksichtigt werden.

Nach Durchführen des Algorithmus ergibt sich aber, dass $s_1 \sim s_5 \sim s_6$ gilt (siehe Tabelle unten). Also ändert sich die Anzahl an Zuständen auch hier nicht.

Zustandsübergangstabelle:

δ	0	1
s_0	s_1	s_0
s_1	s_2	s_1
s_2	s_3	s_1
s_3	s_4	s_5
s_4	s_1	s_0
s_5	s_2	s_5
s_6	s_2	s_1

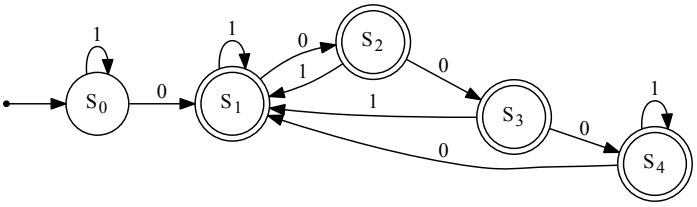
Minimierungstabelle:

s_1	$\times_0$					
s_2	$\times_0$	$\times_3$				
s_3	$\times_0$	$\times_2$	$\times_2$			
s_4	$\times_0$	$\times_1$	$\times_1$	$\times_1$		
s_5	$\times_0$	$-$	$\times_3$	$\times_2$	$\times_1$	
s_6	$\times_0$	$-$	$\times_3$	$\times_2$	$\times_1$	$-$
	s_0	s_1	s_2	s_3	s_4	s_5

c) Wie viele Zustände hätte der minimierte Automat, wenn sich δ zu folgendem Zustandsdiagramm δ_{neu} ändern würde (die einzige Änderung im Vergleich zur minimierten Version des Automaten aus a) ist, dass es keinen Übergang mehr von Zustand s_4 zu Zustand s_0 gibt und stattdessen von s_4 über 1 nach s_4 übergegangen wird)? Geben sie den minimierten Automaten vollständig an. (Wenn Sie "scharf hinsehen", ist kein Algorithmus erforderlich.)

$$A_{neu} = (\{0,1\}, \{s_0, s_1, s_2, s_3, s_4\}, \delta_{neu}, s_0, \{s_1, s_2, s_3, s_4\})$$

δ_{neu}:

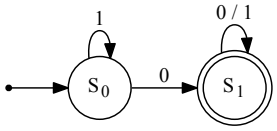


Lösung:

Der minimierte Automat hätte nur 2 Zustände, da man vom Endzustand s_1 nur in andere Endzustände kommt. Damit ergibt sich der minimale Automat zu:

$$A'' = (E, S'', \delta'', s_0, F'') \text{ mit } E = \{0,1\},\ S'' = \{s_0, s_1\},\ F'' = \{s_1\}$$

δ'':



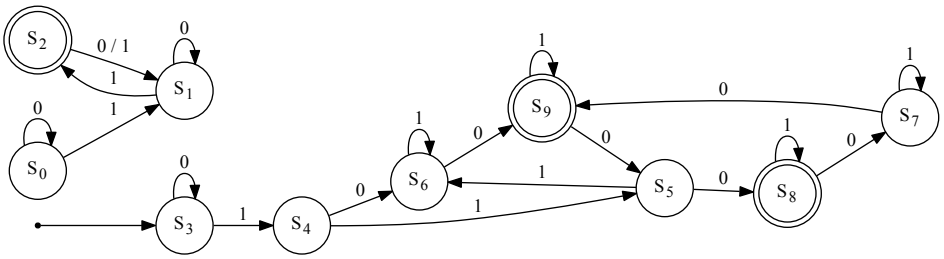
Aufgabe 27 ★★⋆

MIN-AC **Minimierung endlicher Automaten**

Gegeben sei der endliche Automat

$$A = (\{0,1\}, \{s_0, \ldots, s_9\}, \delta, s_3, \{s_2, s_8, s_9\})$$

δ:



a) Minimieren Sie A und geben Sie den minimierten Automaten vollständig an.

Lösung:

Die Zustände s_0, s_1 und s_2 werden nie erreicht und müssen vor der Minimierung entfernt werden. Der dadurch entstandene Automat wird "vereinfachter Automat" genannt und er wird im Folgenden ausschließlich für die Minimierung betrachtet.

Zustandsübergangstabelle:

δ	0	1
s_3	s_3	s_4
s_4	s_6	s_5
s_5	s_8	s_6
s_6	s_9	s_6
s_7	s_9	s_7
s_8	s_7	s_8
s_9	s_5	s_9

Minimierungstabelle:

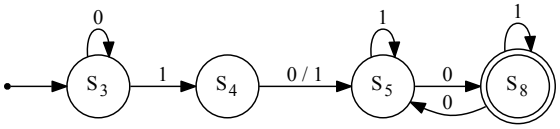
s_4	$\times_2$					
s_5	$\times_1$	$\times_1$				
s_6	$\times_1$	$\times_1$	$-$			
s_7	$\times_1$	$\times_1$	$-$	$-$		
s_8	$\times_0$	$\times_0$	$\times_0$	$\times_0$	$\times_0$	
s_9	$\times_0$	$\times_0$	$\times_0$	$\times_0$	$\times_0$	$-$
	s_3	s_4	s_5	s_6	s_7	s_8

$\Rightarrow s_5 \sim s_6 \sim s_7$ und $s_8 \sim s_9$

Im minimierten Automaten A' gilt: $s_5 \widehat{=} \{s_5, s_6, s_7\}$, $s_8 \widehat{=} \{s_8, s_9\}$

$$A' = (\{0, 1\}, \{s_3, s_4, s_5, s_8\}, \delta', s_3, \{s_8\})$$

δ':



b) Geben Sie einen regulären Ausdruck α an mit $L(\alpha) = L(A) = L(A')$.

Lösung:

$\alpha = 0^\star 1(0 + 1)1^\star 0(1^\star + 01^\star 0)^\star$

c) Geben Sie die Mengen der Zustände des **vereinfachten** (nicht minimierten) Automaten an, die zueinander 0-, 1-, 2- und 3-äquivalent sind.

Lösung:

- Mengen 0-äquivalenter Zustände: $\{s_8, s_9\}, \{s_3, s_4, s_5, s_6, s_7\}$
- Mengen 1-äquivalenter Zustände: $\{s_8, s_9\}, \{s_3, s_4\}, \{s_5, s_6, s_7\}$
- Mengen 2-äquivalenter Zustände: $\{s_8, s_9\}, \{s_3\}, \{s_4\}\{s_5, s_6, s_7\}$
- Mengen 3-äquivalenter Zustände: $\{s_8, s_9\}, \{s_3\}, \{s_4\}\{s_5, s_6, s_7\}$
  (Das entspricht auch den Mengen äquivalenter Zustände.)

4 Rechtslineare Grammatiken und reguläre Ausdrücke

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=349

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 28 ⋆

END-BA **Chomsky-Grammatiken**

Das Konzept der "Sprachen" wird in der Informatik und auch in der Linguistik genutzt, um "Wörter" (oder ling. "Sätze") nach ihrer syntaktischen Korrektheit zu unterscheiden oder um ihre Bedeutung zu definieren und zu extrahieren.

a) Wie lassen sich Sprachen grundsätzlich unterteilen?

Lösung:

Natürliche Sprachen:

- z. B. Deutsch, Englisch, Französisch, ...

Formale Sprachen:

- Programmiersprachen: z. B. Java
- Spezifikationssprachen: z. B. UML
- Anfragesprachen: z. B. SQL
- Mark-up-Sprachen: z. B. HTML

b) Was gehört zur Beschreibung einer Sprache?

Lösung:

- **Syntax**: Menge von Vorschriften/Regeln, die den formal korrekten Aufbau von Wörtern (Sätzen) angeben (beispielsweise Chomsky-Grammatiken).
- **Semantik**: Definition der Bedeutung eines syntaktisch korrekten Wortes (Satzes). Für eine natürliche Sprache ist die Semantik meist komplex und schwer anzugeben. Für eine Programmiersprache ist eine klar definierte Semantik dagegen eine Grundvoraussetzung, damit sie sinnvoll genutzt werden kann.
- **Pragmatik**: Subjektive Aspekte individueller Benutzer/Sprecher/Umstände, z. B. Umgebung, Zeit usw. Die Pragmatik wird vor allem in der Linguistik/Computerlinguistik erforscht und spielt in der reinen Informatik normalerweise keine Rolle.

c) Was ist eine Chomsky-Grammatik? Erläutern Sie die einzelnen Komponenten kurz.

Lösung:

Eine Chomsky-Grammatik definiert eine Sprache und wird beschrieben durch ein Tupel $G = (N, T, P, S)$ mit

- N : Menge der Nonterminalsymbole (syntaktische Variablen)
- T : Menge der Terminalsymbole (syntaktische Konstanten)
- $P \subseteq (N \cup T)^+ \backslash T^\star \times (N \cup T)^\star$: Menge der Produktionen
- $S \in N$: Startsymbol

N, T und P sind jeweils endlich und nicht leer.

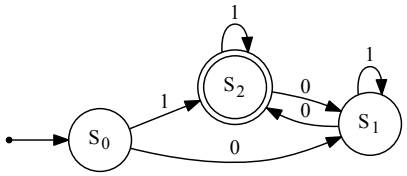
Aufgabe 29 ★

REC-AA **Rechtslineare Grammatiken und endliche Automaten**

Gegeben sei folgender endlicher Automat

$$A = (\{0, 1\}, \{s_0, s_1, s_2\}, \delta, s_0, \{s_2\})$$

δ:



a) Um welchen Automatentyp (deterministisch oder nichtdeterministisch) handelt es sich bei dem abgebildeten Automaten?

Lösung:

Deterministisch, da es von jedem Zustand aus mit jedem Eingabezeichen genau einen Übergang gibt **und** nichtdeterministisch, denn nichtdeterministische Automaten sind eine Obermenge von deterministischen.

b) Welche Sprache $L(A)$ akzeptiert der Automat?

Lösung:

Sprache der nichtleeren Wörter, die eine gerade Anzahl an Nullen enthalten:

$$L(A) = \{w \in \{0, 1\}^\star \mid |w| > 0 \text{ und } |w|_0 \bmod 2 = 0\}$$

c) Geben Sie eine rechtslineare Grammatik G an, die die Sprache $L(A)$ erzeugt. Es soll also gelten $L(A) = L(G)$.

Lösung:

Die Grammatik G lautet:

$$\begin{aligned} G &= (N, T, P, S) \\ N &= \{S, A\}, \\ T &= \{0, 1\}, \\ P &= \{S \rightarrow 0A \mid 1S \mid 1, \\ &\quad A \rightarrow 0S \mid 1A \mid 0\} \end{aligned}$$

Aufgabe 30 ★

REC-AB **Rechtslineare Grammatiken**

Geben Sie eine rechtslineare Grammatik G an, sodass gilt $L(G) = L$ mit

$$L = \{w \in \{0,1\}^+ \mid \forall u, v \in \{0,1\}^\star : w \neq u1111v\}$$

Wörter aus L sind also mindestens ein Zeichen lang und dürfen die Teilfolge 1111 nicht enthalten. Produzieren Sie das Testwort: 011100110001110.

Lösung:

Die Grammatik G lautet:

$$\begin{aligned} G &= (N, T, P, S), \\ N &= \{S, A, B, C\}, \\ T &= \{0, 1\}, \\ P &= \{S \rightarrow 0S \mid 1A \mid 1 \mid 0, \\ &\quad A \rightarrow 0S \mid 1B \mid 1 \mid 0, \\ &\quad B \rightarrow 0S \mid 1C \mid 1 \mid 0, \\ &\quad C \rightarrow 0S \mid 0 \mid \lambda\} \end{aligned}$$

Produktion des Testwortes:

$$\begin{aligned} &S \Rightarrow 0S \Rightarrow 01A \Rightarrow 011B \Rightarrow 0111C \Rightarrow 01110S \Rightarrow 011100S \Rightarrow \\ &0111001A \Rightarrow 01110011B \Rightarrow 011100110S \Rightarrow 0111001100S \Rightarrow 01110011000S \Rightarrow \\ &011100110001A \Rightarrow 01110011000011B \Rightarrow 011100110000111C \Rightarrow 011100110001110 \end{aligned}$$

Aufgabe 31 ⋆⋆

REC-AC **Rechtslineare Grammatiken**

Gegeben sei die Sprache

$$L = \{w \in \{0,1\}^+ \mid \forall u, v \in \{0,1\}^\star : w = u1v \Rightarrow \exists x, y \in \{0,1\}^\star : \\ (u = x1 \wedge v = 00y) \vee v = 100y\}$$

Wörter aus L müssen also nichtleer sein und jede 1 muss im Rahmen der Zeichenfolge 1100 auftreten; Beispiele: $0001100 \in L$, $01101 \notin L$.

a) Geben Sie eine rechtslineare Grammatik G an, sodass gilt $L(G) = L$.

Lösung:

Die Grammatik G lautet:

$$\begin{aligned} G &= (N, T, P, S), \\ N &= \{S, A, B, C\}, \\ T &= \{0, 1\}, \\ P &= \{S \to 0 \mid 0S \mid 1A, \\ &\quad A \to 1B, \\ &\quad B \to 0C, \\ &\quad C \to 0S \mid 0\} \end{aligned}$$

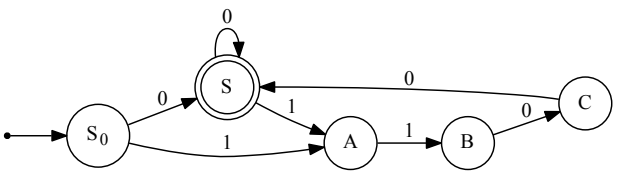
b) Existiert ein deterministischer endlicher Automat A zu dieser rechtslinearen Grammatik mit $L(A) = L(G) = L$? Falls ja, konstruieren Sie diesen und geben Sie ihn vollständig an.

Lösung:

Zu jeder rechtslinearen Grammatik existiert auch ein deterministischer endlicher Automat und umgekehrt. Direkt aus der Grammatik lässt sich eine nichtdeterministische Version des Automaten ableiten (beachten Sie die analoge Konstruktion):

$$A_{ndet} = (\{0,1\}, \{S_0, S, A, B, C\}, \delta_{ndet}, S_0, \{S\})$$

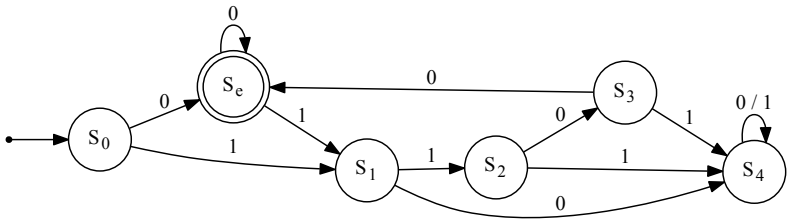
δ_{ndet}:



Daraus kann man leicht den deterministischen Automaten berechnen, indem man die fehlenden Zustandsübergänge in den Senkenzustand s_4 hinzufügt (die Zustände sind hier wieder mnemonisch mit s_i benannt).

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

δ:



Aufgabe 32 ★⋆

END-AP **Rechtslineare Grammatiken**

Gegeben seien die Sprachen $L_i, i \in \{1,2\}$.

$$L_1 = \{w \in \{0,1\}^\star \mid |w|_0 \bmod 2 = 0 \text{ und } |w|_1 \bmod 2 = 1\}$$

$$L_2 = \{w \in \{0,1\}^\star \mid \forall u, v \in \{0,1\}^\star : w \neq u00v, w \neq u11v\}$$

a) Geben Sie jeweils eine rechtslineare Grammatik G_i an, so dass gilt $L(G_i) = L_i$. Geben Sie diese vollständig an.

Lösung:

Die Sprache L_1 enthält alle Wörter $w \in \{0,1\}^\star$ deren Anzahl an 0en gerade und deren Anzahl an 1en ungerade ist. Die Sprache L_2 enthält alle Wörter $w \in \{0,1\}^\star$, die weder die Zeichenfolge 00 noch 11 enthalten. Die entsprechenden rechtslinearen Grammatiken lauten:

$G_1 = \{N, T, P, S\}$
$N = \{S, A, B, C\}$
$T = \{0, 1\}$
$P = \{S \to 0B \mid 1A,$
$A \to 0C \mid 1S \mid \lambda,$
$B \to 0S \mid 1C,$
$C \to 0A \mid 1B\}$

$G_2 = \{N, T, P, S\}$
$N = \{S, A, B\}$
$T = \{0, 1\}$
$P = \{S \to 0A \mid 1B \mid \lambda,$
$A \to 1B \mid \lambda,$
$B \to 0A \mid \lambda\}$

Alternative für G_2 (nicht rechtslinear):
$P = \{S \to 1A \mid A,$
$A \to 01A \mid \lambda \mid 0\}$
$P = \{S \to A \mid B,$
$A \to 01A \mid 0 \mid \lambda\}$
$B \to 10B \mid 1 \mid \lambda\}$

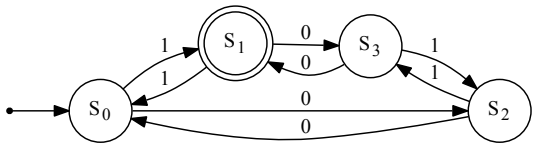
b) Geben Sie jeweils einen endlichen Automaten A_i an mit $L(A_i) = L_i$ erkennen. Geben Sie die Automaten vollständig an.

Lösung:

Der erste Automat lautet wie folgt:

$$A_1 = (\{0, 1\}, \{s_0, s_1, s_2, s_3\}, \delta_1, s_0, \{s_1\})$$

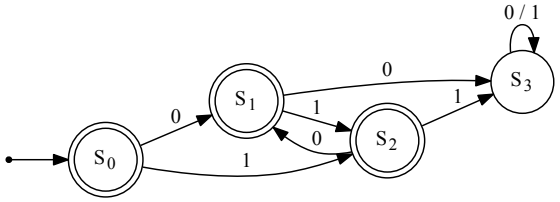
δ_1:



Der zweite Automat lautet wie folgt:

$$A_2 = (\{0, 1\}, \{s_0, s_1, s_2, s_3\}, \delta_2, s_0, \{s_0, s_1, s_2\})$$

δ_2:



Aufgabe 33 ★

REC-AK **Rechslineare Grammatiken und reguläre Ausdrücke**

Gegeben sei die Sprache L aller Wörter über dem Alphabet $E = \{0, 1\}$, die nicht mit einer Eins enden und bei denen die Anzahl aller vorkommenden Einsen ungerade ist:

$$L = \{w \in E^\star \mid |w|_1 \bmod 2 = 1 \text{ und } \forall u \in E^\star : w \neq u1\}$$

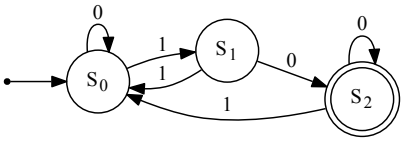
a) Geben Sie einen endlichen Automaten A an, sodass gilt $L(A) = L$. Geben Sie den Automaten vollständig an.

Lösung:

Der endliche Automat A lautet:

$$A = (\{0, 1\}, \{s_0, s_1, s_2\}, \delta, s_0, \{s_2\})$$

δ:



b) Geben Sie eine Grammatik G an, sodass gilt $L(G) = L$. Definieren Sie die Grammatik vollständig.

Lösung:

Die rechtslineare Grammatik G ergibt sich zu:

$$\begin{aligned} G &= (\{A, B, C\}, E, P, A) \\ P &= \{A \to 1B \mid 0A, \\ &\quad B \to 0C \mid 1A \mid 0, \\ &\quad C \to 0C \mid 1A \mid 0\} \end{aligned}$$

Die dritte Zeile kann auch weggelassen werden, wenn in der zweiten Zeile C in B unbenannt wird.

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

Lösung:

Aus der Sprachdefinition abgeleitet:

$$\alpha = 0^\star 1(0^\star 10^\star 10^\star)^\star 00^\star$$

Weitere, aus dem Automaten direkt abgelesene Alternativen (deutlich komplizierter):

$$\alpha = 0^\star 1(10^\star 1)^\star 0(0 + 10^\star 1(10^\star 1)^\star 0)^\star$$

oder

$$\alpha = (0^\star + 11)^\star 10(0 + 1(0^\star + 11)^\star 10)^\star$$

Aufgabe 34 ⋆

END-AO **Rechtslineare Grammatiken und reguläre Ausdrücke**

Gegeben sei die folgende Sprache L

$$L = \{w \in \{a, b\}^\star \mid w = a^k \text{ oder } w = b^l \text{ für } k, l \in \mathbb{N}_0\}.$$

a) Geben Sie eine rechtslineare Grammatik G an, so dass gilt $L(G) = L$. Geben Sie G vollständig an.

Lösung:

Die rechtslineare Grammatik G ergibt sich zu:

$$\begin{aligned} G &= (N, T, P, S) \\ N &= \{S, A, B\} \\ T &= \{a, b\} \\ P &= \{S \rightarrow aA \mid bB \mid \lambda, \\ &\quad A \rightarrow aA \mid a \mid \lambda, \\ &\quad B \rightarrow bB \mid b \mid \lambda\} \end{aligned}$$

b) Geben Sie einen äquivalenten regulären Ausdruck α an, so dass gilt $L(\alpha) = L$.

Lösung:

Der reguläre Ausdruck α lautet:

$$\alpha = a^\star + b^\star$$

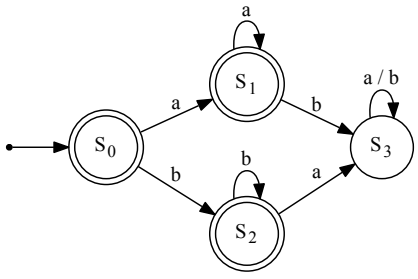
c) Geben Sie einen deterministischen endlichen Automaten $A = (E, S, \delta, s_0, F)$ an mit $L(A) = L$.

Lösung:

Der deterministische endliche Automat A lautet:

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

δ:



Aufgabe 35 ⋆

REC-AE **Reguläre Ausdrücke und endliche Automaten**

Gegeben seien die Sprachen L_1 und L_2 (siehe unten). Entwerfen Sie reguläre Ausdrücke α_1, α_2 und deterministische endliche Automaten A_1, A_2, sodass gilt:

$$L(\alpha_1) = L(A_1) = L_1 \text{ und } L(\alpha_2) = L(A_2) = L_2$$

a) L_1 ist die Sprache der Wörter, bei denen auf jede 1 mindestens zweimal 0 folgt:

$$L_1 = \{w \in \{0,1\}^\star \mid \forall u, v \in \{0,1\}^\star : w = u1v \Rightarrow \exists x \in \{0,1\}^\star : v = 00x\}$$

Lösung:

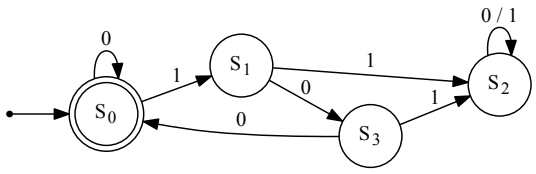
Regulärer Ausdruck:

$$\alpha_1 = 0^\star(1000^\star)^\star 0^\star \text{ oder } \alpha_1 = (0^\star(100)^\star)^\star$$

Endlicher Automat:

$$A_1 = (\{0,1\}, \{s_0, s_1, s_2, s_3\}, \delta_1, s_0, \{s_0\})$$

δ_1:



b) L_2 ist die Sprache der nichtleeren Wörter, deren Länge ein Vielfaches von drei ist:

$$L_2 = \{w \in \{0,1\}^{3n} \mid n \in \mathbb{N}^+\}$$

Lösung:

Regulärer Ausdruck:

$$\alpha_2 = (0+1)(0+1)(0+1)((0+1)(0+1)(0+1))^\star$$

oder

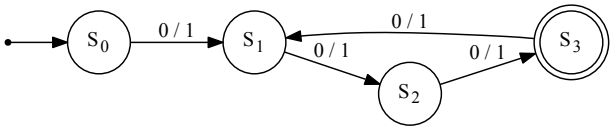
$$\alpha_2 = ((0+1)(0+1)(0+1))^+$$

mit $x^+ =_{def} xx^\star$ ("x^+" ist nicht Teil der Sprache der regulären Ausdrücke und muss definiert werden).

Endlicher Automat:

$$A_2 = (\{0, 1\}, \{s_0, s_1, s_2, s_3\}, \delta_2, s_0, \{s_3\})$$

δ_2:



Aufgabe 36 ★⋆

REC-AF **Reguläre Ausdrücke und endliche Automaten**

Gegeben sei der reguläre Ausdruck $\alpha \in RA(\{a, b\})$ durch

$$\alpha = (b + ab)^\star aa(abb + b)^\star$$

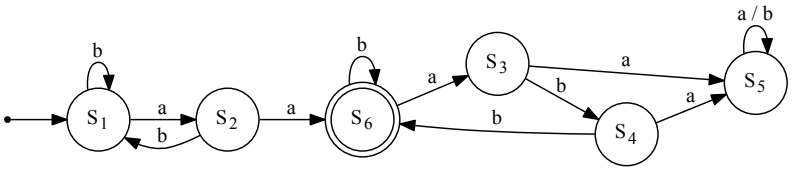
a) Konstruieren Sie einen deterministischen endlichen Automaten A mit $L(A) = L(\alpha)$. Geben Sie A vollständig an.

Lösung:

Bei dieser Sprache können sich am Anfang b und ab beliebig oft abwechseln. Danach muss zweimal das Zeichen a kommen. Dann können noch beliebig viele a's und b's kommen, hinter jedem a müssen aber noch zwei b's kommen. Alle anderen Wörter müssen in einen Senkenzustand führen. Es ergibt sich folgender Automat:

$$A = (\{a, b\}, \{s_1, s_2, s_3, s_4, s_5, s_6\}, \delta, s_1, \{s_6\})$$

δ:



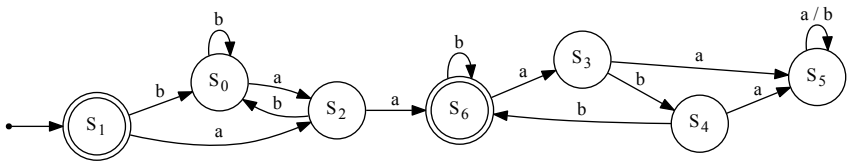
b) Wie sieht ein deterministischer endlicher Automat A' aus, wenn zusätzlich noch das leere Wort in der Sprache enthalten sein soll (sich aber an der Sprache sonst nichts ändert)?

Lösung:

Der Anfangszustand muss nun auch ein Endzustand sein. Zu beachten ist, dass der Zustand s_0 neu hinzugekommen ist, damit z. B. *ab* nicht auch akzeptiert wird.

$$A' = (\{a, b\}, \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}, \delta', s_1, \{s_1, s_6\})$$

δ':



Aufgabe 37 ★

REC-AH **Reguläre Ausdrücke**

Geben Sie zu den folgenden Sprachen L_1, L_2 reguläre Ausdrücke α_1 und α_2 an, sodass

$$L(\alpha_1) = L_1 \text{ und } L(\alpha_2) = L_2$$

a) L_1 ist die Sprache aller Wörter, die nach jeder 0 **genau eine** 1 enthalten:

$$L_1 = \{w \in \{0,1\}^\star \mid \forall u, v \in \{0,1\}^\star : w = u0v \Rightarrow \exists x \in \{0,1\}^\star : v = 10x \vee v = 1\}$$

Lösung:

$\alpha_1 = 1^\star(01)^\star$

b) L_2 ist die Sprache aller Wörter, die nach jeder 0 **mindestens eine** 1 enthalten:

$$L_2 = \{w \in \{0,1\}^\star \mid \forall u, v \in \{0,1\}^\star : w = u0v \Rightarrow \exists x \in \{0,1\}^\star : v = 1x\}$$

Lösung:

$\alpha_2 = (1^\star(011^\star)^\star)^\star$

Aufgabe 38 ⋆⋆

REC-AJ **Reguläre Ausdrücke**

Erzeugen Sie zu den folgenden vier Sprachen L_i mit $(i = a, b, c, d)$ jeweils einen regulären Ausdruck α_i, sodass gilt $L(\alpha_i) = L_i$.

a) Sprache der Wörter mit höchstens 6 Zeichen:

$$L_a = \{w \in \{0,1\}^\star \mid |w| \leq 6\}$$

Lösung:

$\alpha_a = (\emptyset^\star + 0 + 1)(\emptyset^\star + 0 + 1)(\emptyset^\star + 0 + 1)(\emptyset^\star + 0 + 1)(\emptyset^\star + 0 + 1)(\emptyset^\star + 0 + 1)$

b) Sprache der Wörter, die niemals zwei oder mehr Nullen hintereinander enthalten:

$$L_b = \{w \in \{0,1\}^\star \mid \forall u, v \in \{0,1\}^\star : w \neq u00v\}$$

Lösung:

$\alpha_b = (1^\star(01)^\star)^\star(\emptyset^\star + 0)$ oder $\alpha_b = (1^\star + 01)^\star(\emptyset^\star + 0)$

c) Sprache der Wörter, die niemals drei oder mehr Nullen hintereinander enthalten:

$$L_c = \{w \in \{0,1\}^\star \mid \forall u, v \in \{0,1\}^\star : w \neq u000v\}$$

Lösung:

$\alpha_c = (1 + 01 + 001)^\star(\emptyset^\star + 0 + 00)$

d) Sprache der Wörter mit gerader Anzahl an Zeichen, die niemals zwei gleiche Zeichen hintereinander enthalten:

$$L_d = \{w \in \{0,1\}^{2n} \mid n \in \mathbb{N}_0; \forall u, v \in \{0,1\}^\star, \forall x \in \{0,1\} : w \neq uxxv\}$$

Lösung:

$\alpha_d = (01)^\star + (10)^\star$

Aufgabe 39 ⋆⋆

REC-AI **Reguläre Ausdrücke**

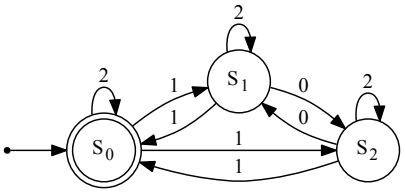
Gegeben seien die unten angegebenen nichtdeterministischen endlichen Automaten A_1, A_2 und A_3. Geben Sie reguläre Ausdrücke α_1, α_2 und α_3 an, sodass gilt:

$$L(\alpha_1) = L(A_1),$$
$$L(\alpha_2) = L(A_2),$$
$$L(\alpha_3) = L(A_3)$$

a) Automatendefinition:

$$A_1 = (\{0, 1, 2\}, \{s_0, s_1, s_2\}, \delta_1, s_0, \{s_0\})$$

δ_1:



Lösung:

Die Zustände s_1 und s_2 sind symmetrisch zueinander aufgebaut. Daher ergeben sich bspw. folgende reguläre Ausdrücke:

$$\alpha_1 = (2^\star + 12^\star(02^\star)^\star 1)^\star \text{ oder } \alpha_1 = (2^\star(1(0 + 2)^\star 1)^\star)^\star$$

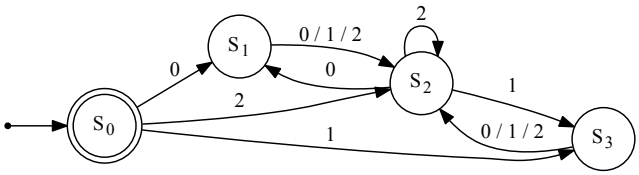
Bei genauerer Betrachtung des Automaten erkennt man, dass s_1 und s_2 sogar äquivalent sind und sich dieser also vereinfachen lässt. Daraus ergibt sich der folgende weiter vereinfachte reguläre Ausdruck:

$$\alpha_1 = (2 + 1(0 + 2)^\star 1)^\star$$

b) Automatendefinition:

$$A_2 = (\{0, 1, 2\}, \{s_0, s_1, s_2, s_3\}, \delta_2, s_0, \{s_0\})$$

δ_2:



Lösung:

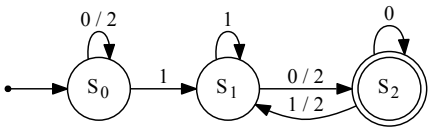
Sobald irgendein Zeichen eingelesen wird, kann kein Endzustand mehr erreicht werden. Für das leere Wort verbleibt man aber im Endzustand. Es ergibt sich folgender regulärer Ausdruck, der die Sprache bezeichnet, die nur das leere Wort enthält:

$$\alpha_2 = \emptyset^\star$$

c) Automatendefinition:

$$A_3 = (\{0, 1, 2\}, \{s_0, s_1, s_2\}, \delta_3, s_0, \{s_2\})$$

δ_3:



Lösung:

Der reguläre Ausdruck α_3 ergibt sich zu:

$$\alpha_3 = (0 + 2)^\star 11^\star (0 + 2)((1 + 2)1^\star (0 + 2) + 0)^\star$$

oder:

$$\alpha_3 = (0 + 2)^\star 11^\star (0 + 2)(0^\star ((1 + 2)1^\star (0 + 2))^\star)^\star$$

Aufgabe 40 ★★

REC-AG **Reguläre Ausdrücke**

Geben Sie einen regulären Ausdruck für die Sprache L der deutschen Festnetznummern an. Dabei soll gelten:

- die Vorwahl besteht aus 4 oder 5 Ziffern und startet mit einer “0”, die 2. Ziffer darf keine “0” sein,
- die Rufnummer besteht aus 4 bis 8 Ziffern und darf nicht mit einer “0” beginnen,
- Vorwahl und Rufnummer werden durch “/” voneinander getrennt.

Lösung:

Alphabet $E = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, /\}$.

$$\begin{aligned}\alpha = {} & 0(1 + 2 + \ldots + 8 + 9)(0 + 1 + 2 + \ldots + 8 + 9) \\ & (0 + 1 + 2 + \ldots + 8 + 9)(0 + 1 + 2 + \ldots + 8 + 9 + \emptyset^\star)/ \\ & (1 + 2 + \ldots + 8 + 9)(0 + 1 + 2 + \ldots + 8 + 9)(0 + 1 + 2 + \ldots + 8 + 9) \\ & (0 + 1 + 2 + \ldots + 8 + 9)(0 + 1 + 2 + \ldots + 8 + 9 + \emptyset^\star) \\ & (0 + 1 + 2 + \ldots + 8 + 9 + \emptyset^\star)(0 + 1 + 2 + \ldots + 8 + 9 + \emptyset^\star) \\ & (0 + 1 + 2 + \ldots + 8 + 9 + \emptyset^\star)\end{aligned}$$

oder

$$\alpha = 0NN_0N_0(N_0 + \emptyset^\star)/NN_0N_0N_0(\emptyset + N_0 + N_0N_0 + N_0N_0N_0 + N_0N_0N_0N_0)$$

mit $N_0 =_{def} (0 + 1 + \ldots + 9), N =_{def} (1 + 2 + \ldots + 9)$.

5 Kellerautomaten

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=350

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 41 ★⋆

KEL-AB **Kellerautomaten**

Gegeben sei folgende Sprache L

$$L = \{w \in \{a,b\}^\star \mid w = (a^2b)^n \text{ mit } n \in \mathbb{N}_0\}$$

a) Geben Sie zu dieser Sprache L einen Kellerautomaten A an mit $L(A) = L$. Geben Sie A vollständig an.

Lösung:

Der Kellerautomat A lautet:

$$A = (E, S, K, \delta, s_0, k_0, F) \text{ mit } E = \{a,b\}, S = \{s_0, s_1, s_e\}, K = \{k_0, a\}, F = \{s_e\}$$

$$\begin{aligned}
\delta : (s_0, a, k_0) &\rightarrow (s_0, ak_0) \\
(s_0, \lambda, k_0) &\rightarrow (s_e, k_0) \text{ (nichtdeterministisch)} \\
(s_0, a, a) &\rightarrow (s_1, a) \\
(s_1, b, a) &\rightarrow (s_e, \lambda) \\
(s_e, a, k_0) &\rightarrow (s_0, ak_0)
\end{aligned}$$

Anmerkung: Es könnte hier auch ein Kellerautomaten angeben werden, der ohne Zugriffe auf den Keller auskommt, also ein endlicher Automat, vgl. Teilaufgabe b).

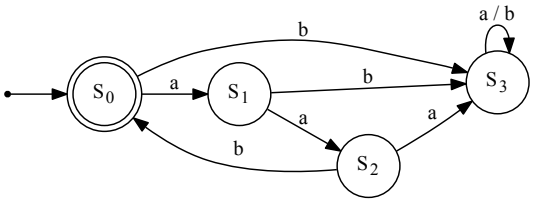
b) Geben Sie für L einen deterministischen endlichen Automaten $A' = (E, S, \delta, s_0, F)$ an mit $L(A') = L$. Geben Sie diesen vollständig an.

Lösung:

Der deterministische endliche Automat A' lautet:

$$A' = (E, S, \delta, s_0, F) \text{ mit } E = \{a,b\}, S = \{s_0, s_1, s_2, s_3\}, F = \{s_0\}$$

δ:



c) Zu welcher/welchen Chomsky-Sprachklasse(n) gehört die Sprache L nach den Ergebnissen aus a) und b)?

Lösung:

Typ 0, Typ 1, Typ 2 und Typ 3

Aufgabe 42 ⋆

KEL-AE **Kellerautomaten**

Gegeben sei folgende Sprache L mit

$$L = \{w \in \{a,b\}^\star \mid w = (ab)^i(ba)^i \text{ mit } i \in \mathbb{N}_0\}$$

a) Geben Sie einen deterministischen Kellerautomaten A an, mit $L(A) = L$. Geben Sie A vollständig an.

Lösung:

Der deterministische Kellerautomat A lautet:

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

$$\begin{aligned}
\delta: \quad (s_0, a, k_0) &\rightarrow (s_1, ak_0) \\
(s_1, b, a) &\rightarrow (s_1, ba) \\
(s_1, a, b) &\rightarrow (s_1, ab) \\
(s_1, b, b) &\rightarrow (s_2, \lambda) \\
(s_2, a, a) &\rightarrow (s_2, \lambda) \\
(s_2, b, b) &\rightarrow (s_2, \lambda) \\
(s_2, \lambda, k_0) &\rightarrow (s_e, k_0)
\end{aligned}$$

b) Zeigen Sie, dass Ihr Kellerautomat das Testwort *abba* akzeptiert.

Lösung:

Erkennung des Testwortes:

$(s_0, abba, k_0) \vdash (s_1, bba, ak_0) \vdash (s_1, ba, bak_0) \vdash (s_2, a, ak_0) \vdash (s_2, \lambda, k_0) \vdash (s_e, k_0)$

Aufgabe 43 ⋆⋆

KEL-AC **Kellerautomaten**

Geben Sie für folgende Sprache L einen deterministischen Kellerautomaten A an mit $L(A) = L$.

$$L = \{w \in \{a, b, \$\}^\star \mid w = u\$u', u \in \{a, b\}^\star\}$$

Definieren Sie den Kellerautomaten A vollständig.

Lösung:

$$A = (E, S, K, \delta, s_0, k_0, F) \text{ mit } E = \{a, b, \$\}, S = \{s_0, s_1, s_2\}, K = \{k_0, a, b\}, F = \{s_2\}$$

$$\begin{aligned}
\delta : (s_0, a, k_0) &\rightarrow (s_0, ak_0) \\
(s_0, b, k_0) &\rightarrow (s_0, bk_0) \\
(s_0, a, a) &\rightarrow (s_0, aa) \\
(s_0, a, b) &\rightarrow (s_0, ab) \\
(s_0, b, a) &\rightarrow (s_0, ba) \\
(s_0, b, b) &\rightarrow (s_0, bb) \\
(s_0, \$, k_0) &\rightarrow (s_1, k_0) \\
(s_0, \$, a) &\rightarrow (s_1, a) \\
(s_0, \$, b) &\rightarrow (s_1, b) \\
(s_1, a, a) &\rightarrow (s_1, \lambda) \\
(s_1, b, b) &\rightarrow (s_1, \lambda) \\
(s_1, \lambda, k_0) &\rightarrow (s_2, k_0)
\end{aligned}$$

Aufgabe 44 ⋆⋆

KEL-AD **Kellerautomaten**

Gegeben sei folgende Sprache L

$$L = \{w \in \{a, b, c\}^\star \mid w = a^m c^q b^n a^p \text{ mit } m = n + p;\ q = 2r;\ m, n, p, r \in \mathbb{N}_0\}$$

Geben Sie zu L einen nichtdeterministischen Kellerautomaten A mit $L(A) = L$ vollständig an.

Lösung:

Die Sprache L enthält alle Wörter w, die am Anfang so viele a's enthalten wie am Ende des Wortes b's und a's stehen. Dazwischen steht eine gerade Anzahl an c's. Zu beachten ist, dass $m, n, p, r \in \mathbb{N}_0$ gilt. Dadurch ergibt sich folgender Kellerautomat:

$$A = (\{a, b, c\}, \{s_0, s_1, s_2, s_3, s_4, s_e\}, \{k_0, a, c\}, \delta, s_0, k_0, \{s_e\})$$

$$\begin{aligned}
\delta: \quad (s_0, \lambda, k_0) &\rightarrow (s_e, k_0) \\
(s_0, a, k_0) &\rightarrow (s_1, ak_0) \\
(s_0, c, k_0) &\rightarrow (s_2, ck_0) \\
(s_1, a, a) &\rightarrow (s_1, aa) \\
(s_1, b, a) &\rightarrow (s_3, \lambda) \\
(s_1, a, a) &\rightarrow (s_4, \lambda) \\
(s_1, c, a) &\rightarrow (s_2, ca) \\
(s_2, c, c) &\rightarrow (s_2, \lambda) \\
(s_2, c, a) &\rightarrow (s_2, ca) \\
(s_2, c, k_0) &\rightarrow (s_2, ck_0) \\
(s_2, \lambda, k_0) &\rightarrow (s_e, k_0) \\
(s_2, b, a) &\rightarrow (s_3, \lambda) \\
(s_2, a, a) &\rightarrow (s_4, \lambda) \\
(s_3, b, a) &\rightarrow (s_3, \lambda) \\
(s_3, \lambda, k_0) &\rightarrow (s_e, k_0) \\
(s_3, a, a) &\rightarrow (s_4, \lambda) \\
(s_4, a, a) &\rightarrow (s_4, \lambda) \\
(s_4, \lambda, k_0) &\rightarrow (s_e, k_0)
\end{aligned}$$

Anmerkung: Durch die Übergänge (s_0, λ, k_0) gekoppelt mit (s_0, a, k_0) bzw. (s_0, c, k_0), den doppelten Übergang von (s_1, a, a) und die Übergänge (s_2, λ, k_0) gekoppelt mit (s_2, c, k_0) wird der Automat nichtdeterministisch.

Aufgabe 45 ★★⋆

KEL-AA **Kellerautomaten**

Gegeben sei folgende Sprache L

$$L = \{w \in \{0,1\}^\star \mid |w|_0 = 2 \cdot |w|_1\}$$

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

Lösung:

Wörter, die zu dieser Sprache gehören, enthalten doppelt so viele Nullen wie Einsen. Folgende Idee liegt dem Kellerautomaten A zugrunde:

- Zustand s_0: Das Verhältnis von Nullen zu Einsen ist genau 2 : 1
- Zustand s_1: Das Verhältnis von Nullen zu Einsen ist größer als 2 : 1 (zu viele Nullen vorhanden)
- Zustand s_2: Übergangszustand von s_1 zu s_3 bzw. Auffangzustand zum Löschen von zwei Nullen bei einer Eins
- Zustand s_3: Das Verhältnis von Nullen zu Einsen ist kleiner als 2 : 1 (zu wenige Nullen vorhanden)

$A = (E, S, K, \delta, s_0, k_0, F)$ mit $E = \{0,1\}, S = \{s_0, s_1, s_2, s_3\}, K = \{0, 1, b, k_0\}, F = \{s_0\}$

$$\begin{aligned}
\delta : (s_0, 0, k_0) &\to (s_1, 0k_0) \\
(s_0, 1, k_0) &\to (s_3, 1k_0) \\
(s_1, 0, 0) &\to (s_1, 00) \\
(s_1, 1, 0) &\to (s_2, \lambda) \\
(s_2, \lambda, 0) &\to (s_1, \lambda) \\
(s_1, \lambda, k_0) &\to (s_0, k_0) \\
(s_2, \lambda, k_0) &\to (s_3, bk_0) \\
(s_3, 1, 1) &\to (s_3, 11) \\
(s_3, 0, 1) &\to (s_3, b) \\
(s_3, 0, b) &\to (s_3, \lambda) \\
(s_3, 1, b) &\to (s_3, b1) \\
(s_3, \lambda, k_0) &\to (s_0, k_0)
\end{aligned}$$

Anmerkung: Beachten Sie, dass der Kellerautomat trotz mehrerer λ-Übergänge deterministisch ist, da es nie die Möglichkeit gibt, in einem bestimmten Zustand entweder λ oder ein anderes Zeichen zu lesen, wenn bei beiden Übergängen das obere Kellerzeichen gleich ist.

b) Zeigen Sie, dass Ihr Kellerautomat das Testwort 100011100000 erkennt.

Lösung:

Erkennung des Testwortes:

$(s_0, 100011100000, k_0) \vdash (s_3, 00011100000, 1k_0) \vdash (s_3, 0011100000, bk_0)$
$\vdash (s_3, 011100000, k_0) \vdash (s_0, 011100000, k_0) \vdash (s_1, 11100000, 0k_0) \vdash (s_2, 1100000, k_0)$
$\vdash (s_3, 1100000, bk_0) \vdash (s_3, 100000, b1k_0) \vdash (s_3, 00000, b11k_0) \vdash (s_3, 0000, 11k_0)$
$\vdash (s_3, 000, b1k_0) \vdash (s_3, 00, 1k_0) \vdash (s_3, 0, bk_0) \vdash (s_3, \lambda, k_0) \vdash (s_0, k_0)$

Aufgabe 46 ★

KEL-AF **Kellerautomaten**

Gegeben sei folgende kontextfreie Grammatik $G = (N, T, P, S)$ mit

$$N = \{S\},$$
$$T = \{a, b\},$$
$$P = \{S \rightarrow aab \mid bbS\}$$

a) Geben Sie einen nichtdeterministischen Kellerautomaten $A = (E, S, K, \delta, s_0, k_0, F)$ an, der zu G äquivalent ist. Definieren Sie A vollständig.

Lösung:

Die Sprache L, die G beschreibt, lautet: $L(G) = \{w \in \{a, b\}^\star \mid w = b^{2n}aab, n \in \mathbb{N}_0\}$

Daraus ergibt sich folgender Kellerautomat:

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

$$\begin{aligned}
\delta : (s_0, b, k_0) &\rightarrow (s_1, k_0) \\
(s_0, a, k_0) &\rightarrow (s_2, k_0) \\
(s_1, b, k_0) &\rightarrow (s_0, k_0) \\
(s_2, a, k_0) &\rightarrow (s_3, k_0) \\
(s_3, b, k_0) &\rightarrow (s_e, k_0)
\end{aligned}$$

Anmerkungen:

- Beachten Sie, dass dieser Kellerautomat den Keller nicht nutzt. Es existiert folglich ein äquivalenter endlicher Automat. Dies folgt aus der Tatsache, dass die Sprache $L(G)$ regulär ist (obwohl G zwei kontextfreie Regeln enthält). Bei Nutzung des Kellers bräuchte man aber u. U. weniger Zustände.
- Der angegebene Kellerautomat ist in diesem Fall deterministisch und somit auch nichtdeterministisch, da die deterministischen Kellerautomaten eine Teilmenge der nichtdeterministischen darstellen.

b) Geben Sie die Konfigurationsübergänge von A für das Testwort *bbbbaab* an.

Lösung:

Erkennung des Testwortes:

$(s_0, bbbbaab, k_0) \vdash (s_1, bbbaab, k_0) \vdash (s_0, bbaab, k_0) \vdash (s_1, baab, k_0) \vdash (s_0, aab, k_0) \vdash (s_2, ab, k_0) \vdash (s_3, b, k_0) \vdash (s_e, k_0)$

6 Kontextfreie Grammatiken

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=351

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 47 ⋆⋆

SPR-AE **Sprachen**

Argumentieren Sie, warum die angegebenen Aussagen zu Sprachen über einem konstanten Alphabet E wahr bzw. falsch sind.

Lösung:

	Wahr	Falsch
Wenn die Sprachen A und B regulär sind, dann ist jede Sprache C mit $A \subseteq C \subseteq B$ auch regulär.	☐	☒

Erklärung: Sei $C = \{0^n 1^n \mid n \in \mathbb{N}\}$ eine nicht-reguläre Sprache. Es gilt $\emptyset \subseteq C \subseteq E^\star$, und damit ist die Aussage falsch.

Sei A regulär und B nicht-regulär. Dann ist $A \cap B$ stets regulär.	☐	☒

Erklärung: Wenn $B \subseteq E^\star$ nicht regulär ist, dann ist $E^\star \cap B = B$ immer noch nicht regulär.

Der Durchschnitt von zwei nicht-regulären Sprachen A und B ist stets nicht-regulär.	☐	☒

Erklärung: Seien z. B. $A = \{0^n 1^n \mid n \in \mathbb{N}\}$ und $B = \{1^n 0^n \mid n \in \mathbb{N}\}$ zwei nicht-reguläre Sprachen. Dann ist $A \cap B = \emptyset$ trotzdem regulär.

Die Menge der kontextfreien Sprachen L_2 ist gegen Komplementbildung abgeschlossen (d. h. $L \in L_2 \Leftrightarrow \overline{L} \in L_2$).	☐	☒

Erklärung: Die Sprache $L = \{w \in \{0,1\}^\star \mid \forall u \in \{0,1\}^\star : w \neq uu\}$ ist kontextfrei, ihr Komplement $\overline{L} = \{w \in \{0,1\}^\star \mid \exists u \in \{0,1\}^\star : w = uu\}$ ist jedoch nicht kontextfrei (siehe Heimübungsblatt 3).

Hinweis: Für ein Alphabet E ist sowohl $E^\star$ als auch $\emptyset$ durch endliche Automaten, rechtslineare Grammatiken usw. darstellbar und somit regulär. Die zugehörigen (nichtdeterministischen) endlichen Automaten bestehen bspw. beide aus nur einem einzigen Zustand, der für $E^\star$ gleichzeitig Endzustand ist und für $\emptyset$ nicht.

Aufgabe 48 ⋆

KON-AA **Kontextfreie Grammatiken**

Geben Sie eine kontextfreie Grammatik G an, für die gilt $L(G) = L$ mit

$$L = \{0^i 1^j \mid i, j \in \mathbb{N}_0, i < j\}$$

Definieren Sie die Grammatik vollständig.

Lösung:

Die Grammatik G lautet:

$$\begin{aligned} G &= (N, T, P, S) \text{ mit} \\ N &= \{S\} \\ T &= \{0, 1\} \\ P &= \{S \to 0S1 \mid S1 \mid 1\} \end{aligned}$$

Aufgabe 49 ⋆

KON-AH **Kontextfreie Grammatiken**

Gegeben sei eine Sprache L von Palindromen der Länge 8:

$$L = \{vv' \mid v \in \{a, b\}^\star, |v| = 4\}$$

Es gilt beispielsweise: $abbbbbba \in L$. Gesucht ist eine Grammatik $G = (N, T, P, S)$, für die gilt: $L(G) = L$.

a) Geben Sie eine solche Grammatik G vollständig an.

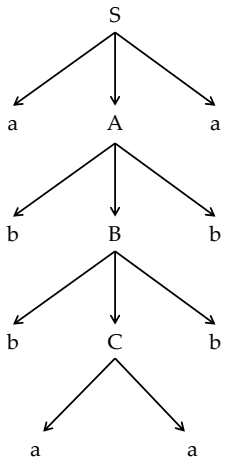
Lösung:

Die Grammatik G lautet:

$$\begin{aligned} G &= (\{S, A, B, C\}, \{a, b\}, P, S), \\ P &= \{S \to aAa \mid bAb, \\ &\quad A \to aBa \mid bBb, \\ &\quad B \to aCa \mid bCb, \\ &\quad C \to aa \mid bb\} \end{aligned}$$

b) Zeichnen Sie einen Ableitungsbaum für das Testwort $abbaabba$.

Lösung:



c) Existiert eine rechtslineare Grammatik G_{rl}, die diese Sprache erzeugt?

Lösung:

Da die Sprache nur endlich viele Wörter enthält, gibt es eine rechtslineare Grammatik G_{rl}, ebenso wie einen endlichen Automaten A und einen regulären Ausdruck α mit $L(G_{rl}) = L(A) = L(\alpha) = L$.

Aufgabe 50 ★

KON-AI **Kontextfreie Grammatiken**

Gegeben seien die folgenden Sprachen L_1 und L_2:

$$L_1 = \{a^{2k+1}b^{2k} \mid k \in \mathbb{N}_0\},$$
$$L_2 = \{(ab)^k c^{2k} \mid k \in \mathbb{N}\}$$

a) Geben Sie eine kontextfreie Grammatik $G_1 = (N_1, T_1, P_1, S_1)$ vollständig an, für die gilt: $L(G_1) = L_1$.

Lösung:

Die kontextfreie Grammatik G_1 lautet:

$$G_1 = (\{S\}, \{a, b\}, P_1, S),$$
$$P_1 = \{S \rightarrow aaSbb \mid a\}$$

b) Geben Sie eine kontextfreie Grammatik $G_2 = (N_2, T_2, P_2, S_2)$ vollständig an, für die gilt: $L(G_2) = L_2$.

Lösung:

Die kontextfreie Grammatik G_2 lautet:

$$G_2 = (\{S\}, \{a, b, c\}, P_2, S),$$
$$P_2 = \{S \rightarrow abScc \mid abcc\}$$

Aufgabe 51 ★⋆

KON-AL **Kontextfreie Grammatiken**

Gesucht ist eine kontextfreie Grammatik G, für die gilt: $L(G) = L$ mit

$$L = \{a^n b^n c^m \mid m, n \geq 0\}.$$

Außerdem sei das Testwort *aaabbbc* gegeben.

a) Geben Sie die Grammatik $G = (N, T, P, S)$ vollständig an.

Lösung:

Es ergibt sich folgende Grammatik:

$$\begin{aligned} G &= (N, T, P, S) \text{ mit} \\ N &= \{S, A, B\}, \\ T &= \{a, b, c\}, \\ P &= \{S \rightarrow AB, \\ &\quad A \rightarrow aAb \mid \lambda, \\ &\quad B \rightarrow cB \mid \lambda\} \end{aligned}$$

b) Geben Sie die Produktion des Testwortes an.

Lösung:

$S \Rightarrow AB \Rightarrow aAbB \Rightarrow aaAbbB \Rightarrow aaaAbbbB \Rightarrow aaabbbB \Rightarrow aaabbbcB \Rightarrow aaabbbc$

Aufgabe 52 ⋆

KON-AJ **Kontextfreie Grammatiken**

Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind und begründen Sie kurz:

Lösung:

	Wahr	Falsch
Jede kontextfreie Sprache ist eine Typ-0-Sprache.	☒	☐

Erklärung: Jede kontextfreie Sprache ist eine Typ-2-Sprache; Typ-2-Sprachen bilden eine Teilmenge der Typ-0-Sprachen.

Kontextfreie Grammatiken sind monoton.	☐	☒

Erklärung: Da Regeln der Form $A \to \lambda$ erlaubt sind, sind kontextfreie Grammatiken nicht monoton.

Die Sprache $L = \{a^n(bc)^n \mid n \in \mathbb{N}_0\}$ kann von einer kontextfreien Grammatik erzeugt werden.	☒	☐

Erklärung: Z. B.: $G = (\{S\}, \{a, b, c\}, \{S \to aSbc \mid \lambda\}, S)$.

Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' in Greibach-Normalform mit $L(G') = L(G)\backslash\{\lambda\}$.	☒	☐

Erklärung: Bis auf Sprachen, die das leere Wort enthalten, können alle Typ-2-Sprachen durch eine Grammatik in Greibach-Normalform erzeugt werden. Beweis: Übung.

Eine Grammatik in Greibach-Normalform ist automatisch auch in Chomsky-Normalform.	☐	☒

Erklärung: Die Greibach-Normalform besteht aus Regeln der Form $A \to a\varphi$, die Chomsky-Normalform aber aus Regeln der Form $A \to a \mid BC$ mit $a \in T; A, B, C \in N; \varphi \in N^\star$. Die beiden Normalformen haben also, bis auf die terminalen Regeln, vollkommen unterschiedlich strukturierte Regeln.

Eine Typ-3-Sprache kann von einer kontextfreien Grammatik erzeugt werden.	☒	☐

Erklärung: Jede rechtslineare Grammatik ist kontextfrei, und jede Typ-3-Sprache kann von einer rechtslinearen Grammatik erzeugt werden.

Aufgabe 53 ★★⋆

KON-AK **Kontextfreie Grammatiken**

Zeigen Sie, dass die Menge der kontextfreien Sprachen bezüglich der Verknüpfung "∩" nicht abgeschlossen ist, dass also der Schnitt zweier kontextfreier Sprachen nicht notwendigerweise kontextfrei ist.

Lösung:

Wähle die folgenden beiden Sprachen:

$$L = \{a^{n_1} b^{n_1} c^{m_1} \mid m_1, n_1 \in \mathbb{N}_0\},$$
$$L' = \{a^{n_2} b^{m_2} c^{m_2} \mid m_2, n_2 \in \mathbb{N}_0\}$$

L enthält also Wörter mit gleich vielen a's und b's, aber beliebig vielen c's ("$a^n b^n c^\star$"), und L' Wörter mit gleich vielen b's und c's, aber beliebig vielen a's ("$a^\star b^m c^m$"). L ist kontextfrei, da die Sprache durch die Grammatik $G = (N, T, P, S)$ erzeugt werden kann, mit

$$\begin{aligned} N &= \{S, A, B\} \\ T &= \{a, b, c\} \\ P &= \{S \to AB, \\ &\quad A \to aAb \mid \lambda, \\ &\quad B \to Bc \mid \lambda\} \end{aligned}$$

L' ist ebenfalls kontextfrei, da die Sprache durch die Grammatik $G' = (N, T, P', S)$ erzeugbar ist, mit

$$\begin{aligned} P' = \{&S \to AB, \\ &A \to Aa \mid \lambda, \\ &B \to bBc \mid \lambda\} \end{aligned}$$

Der Schnitt aus beiden Sprachen ist:

$$\begin{aligned} L \cap L' &= \{a^k b^l c^m \mid \text{ mit } k, l, m, \in \mathbb{N}_0; k = l \text{ (aus } L) \text{ und } l = m \text{ (aus } L')\} \\ &= \{a^n b^n c^n \mid n \in \mathbb{N}_0\} \end{aligned}$$

Um zu zeigen, dass $L \cap L'$ nicht kontextfrei ist, kann man das Pumping-Lemma für kontextfreie Sprachen benutzen. Der Beweis ist eine einfache Übung.

Aufgabe 54 ★⋆

KON-AB **Chomsky-Normalform**

Bringen Sie folgende Grammatik $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{S, A, B, C\} \\
T &= \{a, b, c, d, e\} \\
P &= \{S \to A \mid AB \mid ACa, \\
&\quad A \to b \mid bc, \\
&\quad B \to d, \\
&\quad C \to eCe \mid d\}
\end{aligned}
$$

in Chomsky-Normalform.

Lösung:

(1) Mache G λ-frei – entfällt.

(2) Eliminiere reine Umbenennungen.

$$
\begin{aligned}
G_{(2)} &= (N, T, P_{(2)}, S) \text{ mit} \\
P_{(2)} &= \{S \to b \mid bc \mid AB \mid ACa, \\
&\quad A \to b \mid bc, \\
&\quad B \to d, \\
&\quad C \to eCe \mid d\}
\end{aligned}
$$

(3) Bringe rechte Seiten auf die Form, dass sie entweder aus einem Terminalsymbol oder beliebig viele Nonterminalsymbolen bestehen.

$$
\begin{aligned}
G_{(3)} &= (N_{(3)}, T, P_{(3)}, S) \text{ mit} \\
N_{(3)} &= \{S, A, B, C, D, E, F, G\}, \\
P_{(3)} &= \{S \to b \mid DE \mid AB \mid ACF, \\
&\quad D \to b, \\
&\quad E \to c, \\
&\quad F \to a, \\
&\quad A \to b, \mid DE \\
&\quad B \to d, \\
&\quad C \to GCG \mid d, \\
&\quad G \to e\}
\end{aligned}
$$

(4) Ersetze jede Regel, auf deren rechter Seite mehr als zwei Nonterminalsymbole stehen, durch Regeln, die genau zwei Nonterminalsymbole auf der rechten Seite enthalten.

$$
\begin{aligned}
G_{CNF} &= (N_{(3)} \cup \{H, I\}, T, P_{CNF}, S) \text{ mit} \\
P_{CNF} &= \{S \to b \mid DE \mid AB \mid AH, \\
&\quad H \to CF, \\
&\quad D \to b, \\
&\quad E \to c, \\
&\quad F \to a, \\
&\quad A \to b \mid DE, \\
&\quad B \to d, \\
&\quad C \to GI \mid d, \\
&\quad I \to CG, \\
&\quad G \to e\}
\end{aligned}
$$

Aufgabe 55 ★

KON-AC **Chomsky-Normalform und Cocke-Younger-Kasami-Algorithmus**

Gegeben sei die kontextfreie Grammatik $G = (N, T, P, S)$ mit

$$
\begin{aligned}
N &= \{S, A\} \\
T &= \{a, b\} \\
P &= \{S \to A, \\
&\quad A \to aAb \mid ab\}
\end{aligned}
$$

a) Bringen Sie G in Chomsky-Normalform (CNF).

Lösung:

(1) Mache G λ-frei – entfällt.

(2) Eliminiere reine Umbenennungen. Die einzige Umbenennung, die vorkommt ist $S \to A$.

$$
\begin{aligned}
G_{(2)} &= (\{S\}, T, P_{(2)}, S) \text{ mit} \\
P_{(2)} &= \{S \to aSb \mid ab\}
\end{aligned}
$$

(3) Forme so um, dass die rechte Seite entweder ein Terminalsymbol oder beliebig viele Nonterminalsymbole besitzt.

$$\begin{aligned} G_{(3)} &= (\{S, B, C\}, T, P_{(3)}, S) \text{ mit} \\ P_{(3)} &= \{S \to BSC \mid BC, \\ &\quad B \to a, \\ &\quad C \to b\} \end{aligned}$$

(4) Ersetze jede Regel, auf deren rechter Seite mindestens drei Nonterminalsymbole stehen, durch Regeln mit genau zwei Nonterminalsymbolen auf der rechten Seite.

$$\begin{aligned} G_{CNF} &= (N_{CNF}, T, P_{CNF}, S) \\ N_{CNF} &= \{S, B, C, D\} \\ T &= \{a, b\} \\ P_{CNF} &= \{S \to BD \mid BC, \\ &\quad D \to SC, \\ &\quad B \to a, \\ &\quad C \to b\} \end{aligned}$$

b) Überprüfen Sie mithilfe des Cocke-Younger-Kasami-Algorithmus, ob durch G das Wort $w = aaabbb$ erzeugt werden kann.

Lösung:

Dreieckstabelle:

	a	a	a	b	b	b
$m = 1$	B	B	B	C	C	C
$m = 2$	$-$	$-$	S	$-$	$-$	
$m = 3$	$-$	$-$	D	$-$		
$m = 4$	$-$	S	$-$			
$m = 5$	$-$	D				
$m = 6$	S					

Da das unterste Feld das Symbol S enthält, gilt: $aaabbb \in L(G)$.

Aufgabe 56 ★⋆

KON-AD **Chomsky-Normalform und Cocke-Younger-Kasami-Algorithmus**

Gegeben sei die Grammatik $G = (N, T, P, S)$ mit

$$\begin{aligned} N &= \{S, A, B, C, D\}, \\ T &= \{a, b, c\}, \\ P &= \{S \to ABC \mid AB \mid \lambda, \\ &\quad A \to aC \mid aB \mid b, \\ &\quad B \to bA \mid ABC \mid a, \\ &\quad C \to c \mid D, \\ &\quad D \to CB \mid c\} \end{aligned}$$

a) Bringen Sie G in die Chomsky-Normalform.

Lösung:

(1) G λ-frei machen: Einführen eines neuen Startzustands (wäre hier nicht unbedingt nötig, da der bisherige Startzustand S auf keiner rechten Seite vorkommt).

$$\begin{aligned} G_{(1)} &= (N_{(1)}, T, P_{(1)}, S') \text{ mit} \\ N_{(1)} &= \{S', S, A, B, C, D\}, \\ P_{(1)} &= \{S' \to S \mid \lambda, \\ &\quad S \to ABC \mid AB, \\ &\quad A \to aC \mid aB \mid b, \\ &\quad B \to bA \mid ABC \mid a, \\ &\quad C \to c \mid D, \\ &\quad D \to CB \mid c\} \end{aligned}$$

(2) Reine Umbenennungen (außer $S' \to S$) entfernen: $C \to D$ und $D \to CB$ wird zu $C \to c \mid CB$. Mit $C \to c$ ist auch das ursprüngliche $D \to c$ abgedeckt.

$$\begin{aligned} G_{(2)} &= (N_{(2)}, T, P_{(2)}, S') \text{ mit} \\ N_{(2)} &= \{S', S, A, B, C\}, \\ P_{(2)} &= \{S' \to S \mid \lambda, \\ &\quad S \to ABC \mid AB, \\ &\quad A \to aC \mid aB \mid b, \\ &\quad B \to bA \mid ABC \mid a, \\ &\quad C \to c \mid CB\} \end{aligned}$$

(3) Änderung der rechten Seiten, sodass sie entweder ein Terminalsymbol oder beliebig viele Nonterminalsymbole enthalten:

$$
\begin{aligned}
G_{(3)} &= (N_{(3)}, T, P_{(3)}, S') \text{ mit} \\
N_{(3)} &= \{S', S, A, B, C, E, F\}, \\
P_{(3)} &= \{S' \to S \mid \lambda, \\
&\qquad S \to ABC \mid AB, \\
&\qquad A \to EC \mid EB \mid b, \\
&\qquad B \to FA \mid ABC \mid a, \\
&\qquad C \to c \mid CB, \\
&\qquad E \to a, \\
&\qquad F \to b\}
\end{aligned}
$$

(4) Jede Regel, auf deren rechter Seite mindestens drei Nonterminalsymbole vorkommen, ersetzen durch Regeln mit genau zwei Nonterminalsymbolen:

$$
\begin{aligned}
G_{CNF} &= (N_{CNF}, T, P_{CNF}, S') \text{ mit} \\
N_{CNF} &= \{S', S, A, B, C, E, F, G\}, \\
P_{CNF} &= \{S' \to S \mid \lambda, \\
&\qquad S \to AG \mid AB, \\
&\qquad A \to EC \mid EB \mid b, \\
&\qquad B \to FA \mid AG \mid a, \\
&\qquad C \to c \mid CB, \\
&\qquad E \to a, \\
&\qquad F \to b, \\
&\qquad G \to BC\}
\end{aligned}
$$

b) Überprüfen Sie mit dem Cocke-Younger-Kasami-Algorithmus, ob durch die Grammatik G das Wort $w = acbbacacc$ erzeugt werden kann.

Lösung:

Dreieckstabelle:

	a	c	b	b	a	c	a	c	c
$m=1$	B,E	C	A,F	A,F	B,E	C	B,E	C	C
$m=2$	A,G	$-$	B	S,S'	A,G	C	A,G	$-$	
$m=3$	$-$	C	$-$	S,S',B	S,S',A,G	$-$	$-$		
$m=4$	S,S',A,G	C	S,S'	S,S',B	S,S',B	$-$			
$m=5$	S,S',A,G	$-$	S,S'	S,S',G	G				
$m=6$	S,S',B	$-$	S,S',B	S,S',B					
$m=7$	S,S',B	C	S,S',G						
$m=8$	S,S',A,G	$-$							
$m=9$	S,S',B								

Das Wort ist erzeugbar, da S' in der untersten Zelle enthalten ist.

Anmerkung: Wegen der Regel $S' \to S$ muss in jeder Zelle, die S enthält, auch S' enthalten sein (dies ist eine Ausnahmeregelung, die nicht direkt zum CYK-Algorithmus gehört).

c) Produzieren Sie das Wort $w = acbbacacc$ mithilfe der Grammatiken aus Aufgabenteil a) vor und nach der Umwandlung in CNF.

Lösung:

- Vor Umwandlung in CNF:

$$S \Rightarrow ABC \Rightarrow aCBC \Rightarrow acBC \Rightarrow acABCC \Rightarrow acbBCC \Rightarrow acbABCCC$$
$$\Rightarrow^2 acbbBDCC \Rightarrow acbbaDCC \Rightarrow acbbaCBCC \Rightarrow^\star acbbacacc$$

- Nach Umwandlung in CNF:

$$S' \Rightarrow S \Rightarrow AG \Rightarrow ABC \Rightarrow ECBC \Rightarrow ECBBC \Rightarrow ECFABC$$
$$\Rightarrow ECFAAGC \Rightarrow ECFAECGC \Rightarrow ECFAECBCC \Rightarrow^\star acbbacacc$$

Aufgabe 57 ⋆

KON-AE **Cocke-Younger-Kasami-Algorithmus**

Gegeben sei die Grammatik $G = (N, T, P, S)$ mit $N = \{S, A, B, C\}$, $T = \{a, b\}$ und

$$\begin{aligned} P = \{S &\to AB \mid BC, \\ A &\to BA \mid a, \\ B &\to CC \mid b, \\ C &\to AB \mid a\} \end{aligned}$$

Überprüfen Sie mithilfe des Cocke-Younger-Kasami-Algorithmus, ob $bbaaab \in L(G)$.

Lösung:

Dreieckstabelle:

	b	b	a	a	a	b
$m = 1$	B	B	A, C	A, C	A, C	B
$m = 2$	$-$	S, A	B	B	S, C	
$m = 3$	A	$-$	S, A, C	B		
$m = 4$	$-$	S, A, C	S, C			
$m = 5$	S, A, C	S, C				
$m = 6$	S, C					

Da das Startsymbol im untersten Feld der Tabelle steht, gilt $w \in L(G)$.

Aufgabe 58 ★⋆

KON-AF **Cocke-Younger-Kasami-Algorithmus**

Überprüfen Sie mithilfe des Cocke-Younger-Kasami-Algorithmus, ob das Wort *bcedea* von der Grammatik $G = (N, T, P, S)$ erzeugt werden kann, mit

$$
\begin{aligned}
N &= \{S, A, B, C, D, E, F, G, H, I\} \\
T &= \{a, b, c, d, e\} \\
P &= \{S \to b \mid DE \mid AB \mid AH, \\
&\quad A \to d \mid DE, \\
&\quad B \to d, \\
&\quad C \to GI \mid d, \\
&\quad D \to b, \\
&\quad E \to c, \\
&\quad F \to a, \\
&\quad G \to e, \\
&\quad H \to CF, \\
&\quad I \to CG\}
\end{aligned}
$$

Lösung:

Dreieckstabelle:

	b	c	e	d	e	a
$m = 1$	S, D	E	G	A, B, C	G	F
$m = 2$	S, A	$-$	$-$	I	$-$	
$m = 3$	$-$	$-$	C	$-$		
$m = 4$	$-$	$-$	H			
$m = 5$	$-$	$-$				
$m = 6$	S					

S ist im untersten Feld enthalten, also wird das Wort von der Grammatik erzeugt:

$$bcedea \in L(G)$$

Aufgabe 59 ★⋆

KON-AG **Cocke-Younger-Kasami-Algorithmus**

Gegeben sei die Grammatik $G = (N, T, P, S)$ mit

$$\begin{aligned} N &= \{S, A, B\} \\ T &= \{f, i, n, o\} \\ P &= \{S \to iB \mid nBA, \\ &\quad A \to fAf \mid fBo, \\ &\quad B \to nBo \mid fB \mid \lambda\} \end{aligned}$$

a) Bringen Sie G in Chomsky-Normalform (CNF).

Lösung:

(1) Mache G λ-frei.

$$\begin{aligned} G_{(1)} &= (N, T, P_{(1)}, S) \\ P_{(1)} &= \{S \to iB \mid nBA \mid i \mid nA, \\ &\quad A \to fAf \mid fBo \mid fo, \\ &\quad B \to nBo \mid fB \mid no \mid f\} \end{aligned}$$

(2) Eliminiere reine Umbenennungen; dieser Schritt entfällt hier, da es keine Nonterminal-Umbenennungen gibt.

(3) Forme so um, dass die rechten Seiten entweder aus einem Terminalsymbol oder beliebig vielen Nonterminalsymbolen bestehen.

$$
\begin{aligned}
G_{(3)} &= (N \cup \{I, F, N, O\}, T, P_{(3)}, S) \\
P_{(3)} &= \{S \to IB \mid NBA \mid i \mid NA, \\
&\quad A \to FAF \mid FBO \mid FO, \\
&\quad B \to NBO \mid FB \mid NO \mid f, \\
&\quad I \to i, \\
&\quad F \to f, \\
&\quad N \to n, \\
&\quad O \to o\}
\end{aligned}
$$

(4) Ersetze jede Regel, auf deren rechter Seite mindestens 3 Nonterminalsymbole stehen, durch Regeln, mit genau zwei Nonterminalsymbolen. Es entsteht die Grammatik $G_{CNF} = (N_{CNF}, T, P_{CNF}, S)$ mit

$$
\begin{aligned}
N_{CNF} &= \{S, A, B, C, D, E, I, F, N, O\} \\
T &= \{f, i, n, o\} \\
P_{CNF} &= \{S \to IB \mid NC \mid i \mid NA, \\
&\quad A \to FD \mid FE \mid FO, \\
&\quad B \to NE \mid FB \mid NO \mid f, \\
&\quad C \to BA, \\
&\quad D \to AF, \\
&\quad E \to BO, \\
&\quad I \to i, \\
&\quad F \to f, \\
&\quad N \to n, \\
&\quad O \to o\}
\end{aligned}
$$

b) Überprüfen Sie mithilfe des Algorithmus von Cocke, Younger und Kasami, ob $info \in L(G) = L(G_{CNF})$.

Lösung:

Dreieckstabelle:

	i	n	f	o
$m = 1$	S, I	N	B, F	O
$m = 2$	-	-	A, E	
$m = 3$	-	S, B		
$m = 4$	S			

S ist im letzten Feld enthalten, der Algorithmus liefert also das Ergebnis, dass das angegebene Wort $info$ durch die angegebene Grammatik erzeugt werden kann.

7 Pumping-Lemma

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=352

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 60 ⋆

PUM-AB **Pumping-Lemma für reguläre Sprachen**

Benennen Sie die zentrale Aussage des Pumping-Lemmas für reguläre Sprachen in eigenen Worten.

Lösung:

Das Pumping-Lemma ist eine Aussage über alle Sprachen, die von einem endlichen Automaten (EA) erkannt werden können. Die mit dem Pumping-Lemma eng verbundene Frage ist, ob eine Sprache L von **irgendeinem** EA erkannt werden kann, ob also ein EA A existiert mit $L(A) = L$. Zur Beantwortung dieser Frage geht man folgender Idee nach: Sobald die **Anzahl der Zeichen** eines Wortes w größer als (oder gleich groß wie) die **Anzahl der Zustände** eines EAs ist, der es erkennt, muss es eine Schleife geben, also eine Stelle, an der das Wort "aufgepumpt" werden kann; der EA erkennt dann auch die aufgepumpten Wörter.
Sei $w = xyz$ ein solches Wort, wobei x der Teil von w ist, der sich vor der Schleife befindet, y der Teil in der Schleife und z der Teil nach der Schleife. Dann akzeptiert der Automat nicht nur das Wort xyz, sondern auch die Wörter xz, $xyyz$, $xyyyz$ usw. (allgemein $\forall i : xy^iz$), da man das Wort an der Stelle y aufpumpen kann. Das ist die wesentliche Aussage des Pumping-Lemmas, die für jede Sprache, die von einem EA erkannt werden kann, gilt.
Die Umkehrung dieser Aussage besagt, dass, wenn das Pumping-Lemma für eine Sprache nicht gilt, es auch keinen EA gibt, der diese Sprache erkennt. Diese Einsicht kann verwendet werden, um einen Widerspruchsbeweis zu führen, um zu zeigen, dass es keinen EA für eine Sprache L gibt. Lässt sich zeigen, dass bei einer Zerlegung des Wortes $w \in L$ in drei Teile x, y, z nach den Bedingungen 1) und 2) (siehe Einführung) ein Aufpumpen **nicht** möglich ist, dass es also mindestens ein i gibt, für das $xy^iz \notin L(A)$ (Widerspruch zu Bedingung 3); siehe Einführung), dann kann die Sprache **nicht** von einem endlichen Automaten akzeptiert werden.

Aufgabe 61 ⋆

PUM-AA **Pumping-Lemma für reguläre Sprachen**

Zeigen Sie mithilfe des Pumping-Lemmas für Typ-3-Sprachen (reguläre Sprachen), dass kein endlicher Automat existiert, der die Sprache

$$L = \{w \in \{0,1\}^\star \mid |w|_1 = |w|_0\}$$

erkennt.

Lösung:

Wir nehmen an, L würde von einem endlichen Automaten A erkannt und dieser Automat hätte $|S| = n$ Zustände. Laut Pumping-Lemma existieren dann für jedes Wort $w \in L$ mit $|w| \geq n$ Wörter $x, y, z \in E^\star$ mit $w = xyz$, sodass gilt:

1) $|xy| \leq n$,

2) $|y| \geq 1$,

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

Diese Folgerung des Pumping-Lemmas führen wir im Folgenden zum Widerspruch, um zu zeigen, dass die Annahme, L würde von einem endlichen Automaten erkannt, falsch ist. Dafür müssen wir die Aussage des Pumping-Lemmas umdrehen, also für **ein einziges** Wort w zeigen, dass **alle möglichen** Zerlegungen in x, y und z dazu führen, dass **mindestens eine** der obigen Bedingungen 1), 2) oder 3) nicht erfüllt wird. Wir nehmen dafür an, dass 1) und 2) erfüllt seien und zeigen, dass dann 3) nicht erfüllt sein kann, indem wir **ein einziges** i angeben, dass die dritte Bedingung nicht erfüllt:
Wir wählen das Wort $w = 0^n 1^n \in L$ mit $|w| \geq n$. Wenn $|xy| \leq n$ und $|y| \geq 1$ ist, dann gibt es k, j mit $0 < k \leq j \leq n$, sodass $xy = 0^j$, $y = 0^k$ und $x = 0^{j-k}$, denn xy kann nicht länger als n Zeichen sein und die ersten n Zeichen von w sind Nullen; für das Restwort z gilt dann $z = 0^{n-j}1^n$.
Wenn wir nun beispielsweise $i = 0$ setzen, gilt $w' =_{def} xy^0z = \overbrace{0^{j-k}}^{x}\overbrace{0^{n-j}1^n}^{z} = 0^{n-k}1^n \notin L$, denn wegen $k > 0$ ist die Anzahl der Nullen $n-k$ in w' um mindestens eins kleiner als die Anzahl der Einsen n, was nicht der Definition der Sprache L entspricht. Da nach dem Pumping-Lemma die Bedingung 3) für alle $i \in \mathbb{N}_0$ gelten müsste, sie aber offenbar für $i = 0$ nicht gilt, sind wir zu einem Widerspruch gekommen. Also muss die anfängliche Annahme falsch sein, dass $L = L(A)$ für einen endlichen Automaten A, was beweist, dass die Sprache nicht vom Typ 3 ist.
Bemerkung: Ein Widerspruchsbeweis mit dem Pumping-Lemma beginnt mit der Wahl eines geeigneten Pumpwortes w. Da das Pumping-Lemma eine Aussage über alle Wörter einer bestimmten Länge macht, reichte es in der Umkehrung aus, ein einziges Wort zu finden, für das die Aussage des Pumping-Lemmas **nicht** gilt (allerdings muss dieses Wort variable Länge haben, damit garantiert werden kann, dass es länger als n ist). Wir wählen hier ein möglichst einfach strukturiertes Wort der Sprache, das aber nach n Zeichen eine wesentliche Veränderung in seiner Struktur erfährt, die durch ein Aufpumpen der ersten n Zeichen nicht mitgemacht wird. In diesem Fall enthält das Wort $w = 0^n 1^n$ nur Nullen innerhalb der ersten n Zeichen; pumpt man diese, kommen entweder weitere Nullen hinzu ($i > 1$) oder es werden welche gelöscht ($i = 0$); auf keinen Fall ändert sich aber die Anzahl der Einsen, weswegen sich auf jeden Fall ein Missverhältnis zwischen Nullen und Einsen im gepumpten Wort ergibt. (Jede andere Wahl $i \neq 1$ wäre hier also auch in Ordnung gewesen.) In anderen Fällen kann die Wahl von i schwieriger sein und unter Umständen auch von der jeweiligen konkreten Belegung von x, y und z abhängen. Da diese nicht frei gewählt werden dürfen, muss über alle möglichen

Belegungen (im Rahmen der Pumping-Lemma-Bedingungen) argumentiert werden. Für jede mögliche Belegung darf es aber prinzipiell unterschiedliche Werte für i geben.

Aufgabe 62 ★⋆

PUM-AF **Pumping-Lemma für reguläre Sprachen**

Zeigen Sie, dass folgende Sprache L nicht rechtslinear ist:

$$L = \{a^n bbc^n \mid n \in \mathbb{N}\}$$

Lösung:

Wenn L rechtslinear ist, wird L auch von einem endlichen endlichen Automaten erkannt. Dann besagt das Pumping-Lemma für Typ-3-Sprachen, dass es ein $n \in \mathbb{N}$ geben muss, sodass für alle $w \in L$ mit $|w| \geq n$ eine Zerlegung $w = xyz$ existiert, für die gilt:

1) $|xy| \leq n$,

2) $y \neq \lambda$,

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

Wählen wir nun $w = a^n bbc^n$, gilt $|w| \geq n$. Für jede beliebige Zerlegung, für die 1) und 2) gilt, kann y nur a's enthalten, da $|xy| \leq n$ gilt und die ersten n Zeichen von w alle a sind, und y enthält mindestens ein a wegen $y \neq \lambda$. Genauer gesagt gilt $xy = a^j, y = a^k, x = a^{j-k}$ und $z = a^{n-j}bbc^n$ für $0 < k \leq j \leq n$.

Durch Pumpen mit $i = 0$ entsteht das Wort $w' =_{def} xy^0z = \overbrace{a^{j-k}}^{x}\overbrace{a^{n-j}bbc^n}^{z} = a^{n-k}bbc^n \notin L$ (da wegen $k > 0$ die Anzahl der a's nicht gleich der Anzahl der c's im Wort ist). Damit ergibt sich ein Widerspruch, denn nach dem Pumping-Lemma müsste $w' \in L$ gelten. Wir müssen also die ursprüngliche Annahme aufgeben, es gäbe einen endlichen Automaten, der L akzeptiert, und damit kann L auch nicht rechtslinear sein.

Aufgabe 63 ★⋆

PUM-AC **Pumping-Lemma für reguläre Sprachen**

Für ein Alphabet E und ein Wort $w \in E^\star$ bezeichne $w' \in E^\star$ die Umkehrung von w. Gegeben sei die Sprache

$$L = \{x1x' \mid x \in \{0,1\}^\star\}.$$

a) Zeigen Sie mithilfe des geeigneten Pumping-Lemmas, dass L nicht von einem endlichen Automaten erkannt werden kann.

Lösung:

Angenommen, es existiere ein endlicher Automat (EA) A mit $|S| = n$ Zuständen und $L(A) = L$. Dann gibt es laut Pumping-Lemma für jedes Wort $w \in L$ mit $|w| \geq n$ eine Zerlegung $w = xyz$ mit

1) $|xy| \leq n$,
2) $|y| \geq 1$,
3) $\forall i \in \mathbb{N}_0 : xy^iz \in L$.

Wir wählen als Pumpwort $w = 0^n10^n \in L$, für das offenbar $|w| \geq n$ gilt. Wegen den Bedingungen 1) und 2) existieren j, k mit $0 < k \leq j \leq n$, sodass $xy = 0^j$, $y = 0^k$, $x = 0^{j-k}$ und $z = 0^{n-j}10^n$.

Wird beispielsweise $i = 0$ gewählt, enthält $w' =_{def} xy^0z = \overbrace{0^{j-k}}^{x}\overbrace{0^{n-j}10^n}^{z} = 0^{n-k}10^n$ weniger Nullen im ersten Teil des Wortes als im zweiten Teil; das ergibt sich direkt aus $k > 0$. Also gilt $w' \notin L$, da die Struktur des Wortes nicht der Sprachdefinition entspricht. Damit ergibt sich ein Widerspruch, und wir müssen die ursprüngliche Annahme aufgeben, es gäbe einen EA, der L akzeptiert.

b) Geben Sie eine kontextfreie Grammatik G an mit $L(G) = L$.

Lösung:

Eine Grammatik G mit $L(G) = L$ wäre beispielsweise:

$$\begin{aligned} &G = (N, T, P, S) \text{ mit} \\ &N = \{S\}, \\ &T = \{0, 1\}, \\ &P = \{S \to 1S1 \mid 0S0 \mid 1\} \end{aligned}$$

c) Erzeugen Sie eine Grammatik G' in Greibach-Normalform mit $L(G') = L(G) = L$.

Hinweis: Erlaubt sind dabei nur Produktionen der Form $N \times TN^\star$.

Lösung:

Eine Grammatik G' in Greibach-Normalform mit $L(G') = L$ wäre beispielsweise:

$$\begin{aligned} G &= (N, T, P, S) \\ N &= \{S, A, B\} \\ T &= \{0, 1\} \\ P &= \{S \rightarrow 1SA \mid 0SB \mid 1, \\ &\quad A \rightarrow 1, \\ &\quad B \rightarrow 0\} \end{aligned}$$

Es ist nicht immer so einfach, aus einer beliebigen kontextfreien Grammatik eine Grammatik in Greibach-Normalform zu erzeugen. In diesem Fall haben wir nur die jeweils hintere 1 bzw. 0 der beiden "nicht-Greibach-konformen" Produktionen von G durch ein Zwischen-Nonterminalzeichen A bzw. B ersetzt und dieses in einem zweiten Schritt in 1 bzw. 0 übergehen lassen, um auf G' zu kommen.

Aufgabe 64 ⋆⋆

PUM-AL **Pumping-Lemma für kontextfreie Sprachen**

Gegeben sei die Sprache L mit

$$L = \{a^i b^j c^k \mid i \leq j \leq k \in \mathbb{N}\}$$

Ist L kontextfrei?

Lösung:

Wir nehmen an, L könnte von einer kontextfreien Grammatik G erzeugt werden. Dann existiert (abhängig von der Struktur dieser Grammatik G) eine Konstante $k \in \mathbb{N}$, sodass für alle Wörter $z \in L$ mit $|z| \geq k$ eine Zerlegung $z = uvwxy$ existiert mit

1) $|vwx| \leq k$,
2) $|vx| \geq 1$,
3) $\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$.

Wie bei den Aufgaben zum Pumping-Lemma für reguläre Sprachen, führen wir diese Folgerung des Pumping-Lemmas auch hier zum Widerspruch, um zu zeigen, dass die Annahme, L würde von einer kontextfreien Grammatik erzeugt, falsch ist. Dafür müssen wir die Aussage des Pumping-Lemmas umdrehen, also für **ein einziges** Wort z zeigen, dass **alle möglichen** Zerlegungen dieses Wortes in u, v, w, x und y dazu führen, dass **mindestens eine** der obigen

Bedingungen 1), 2) oder 3) nicht erfüllt wird. Wir nehmen dafür an, dass 1) und 2) erfüllt seien und zeigen, dass dann 3) nicht erfüllt sein kann, indem wir **ein einziges** i angeben, dass die dritte Bedingung nicht erfüllt:
Wir wählen ein Wort $z = a^k b^k c^k \in L$ mit $|z| = 3k \geq k$. Aus $|vwx| \leq k$ und der Struktur von z folgt, dass vx maximal zwei der drei Symbole a, b und c enthalten kann; aus $|vx| \geq 1$ folgt, dass v oder x (oder beide) mindestens eines dieser Symbole enthalten müssen. Es ergeben sich folgende mögliche Fälle:

- ***vx* enthält mindestens ein *a***: dann enthält vx kein c, denn die höchstens k Zeichen von vwx reichen nicht "über die b's hinüber" bis zu den c's im Wort $z = a^k b^k c^k$. Es gilt also:
$$|vx|_a > 0 \Rightarrow |vx|_c = 0$$
Jedes gepumpte Wort $z' =_{def} uv^i wx^i y$ mit $i > 1$ gehört dann nicht zur Sprache L, weil $|z'|_a > |z'|_c$, und das entspricht nicht der Struktur von L.

- ***vx* enthält kein *a***: dann enthält vx mindestens ein b oder ein c, denn es darf ja nicht leer sein. Es gilt also:
$$|vx|_a = 0 \Rightarrow (|vx|_b > 0 \text{ oder } |vx|_c > 0)$$
Dann ist für $i = 0$ das gepumpte Wort $z' =_{def} uv^0 wx^0 y = uwy$ nicht Element der Sprache, weil $|z'|_a > |z'|_b$ oder $|z'|_a > |z'|_c$ gilt und beides nicht der Struktur der Sprache entspricht.

In beiden Fällen entsteht ein Widerspruch, denn laut Pumping-Lemma sollte jedes gepumpte Wort ebenfalls in L liegen. Also müssen wir die ursprüngliche Annahme aufgeben, dass es eine kontextfreie Grammatik G gibt mit $L(G) = L$.
Bemerkung: Wie beim Pumping-Lemma für reguläre Sprachen (vgl. Bemerkung in Lösung zu Aufgabe PUM-AA) steht und fällt auch beim Pumping-Lemma für kontextfreie Sprachen ein Widerspruchsbeweis mit der geschickten Wahl des zu pumpenden Wortes z. Auch hier sollte ein möglichst einfach strukturiertes Wort der Sprache gewählt werden, wobei die Struktur des Wortes diesmal zwei wesentliche Veränderungspunkte haben sollte, die mindestens k Zeichen voneinander entfernt sind. In obigem Beispiel enthält das Wort $z = a^k b^k c^k$ als Trenner zwischen den a's und den c's eine Kette von k b's. Das führt dazu, dass auf keinen Fall a's, b's und c's gleichzeitig gepumpt werden können, da die beiden Pumpstellen höchstens k Zeichen voneinander entfernt sind. Durch geeignete Wahl von i kann dann so gepumpt werden, dass sich die Anzahl der Zeichen im Verhältnis zueinander so verändert, dass das gepumpte Wort nicht mehr in der Sprache liegt, was den gewünschten Widerspruch herbeiführt. Wie man sieht, kann die Wahl von i auch durchaus von der jeweiligen konkreten Belegung der Teilwörter u, v, w, x und y abhängen ($i > 1$ im ersten Fall und $i = 0$ im zweiten Fall).
Im Gegensatz zu den Aufgaben zum Pumping-Lemma für reguläre Sprachen verzichten wir hier auf die genaue Aufschlüsselung der Struktur der Teilwörter u, v, w, x und y. Die Darstellung dieser Struktur ist zwar möglich, aber benötigt oft Fallunterscheidungen und ist komplizierter als im regulären Fall. Da Beweise mit dem Pumping-Lemma für kontextfreie Sprachen ohnehin oft nicht ohne Fallunterscheidungen auskommen, würde die explizite Darstellung der Teilwörter mehr Verwirrung als Einsicht bringen.

Wir verzichten ebenfalls darauf, die Konstante k genau anzugeben und beschränken uns darauf festzustellen, dass es so eine Konstante geben muss. Während beim Pumping-Lemma für reguläre Sprachen die analog verwendete Konstante n einfach der Anzahl der Zustände eines endlichen Automaten entspricht, ist k komplizierter zu bestimmen; es wächst im Wesentlichen exponentiell in der Anzahl der Nonterminalsymbole der Grammatik G. Sowohl für n als auch für k gilt, dass der genaue Wert unwesentlich ist und für die üblichen Beweise mit den Pumping-Lemmata nicht bestimmt werden muss.

Aufgabe 65 ★⋆

PUM-AD **Pumping-Lemma für kontextfreie Sprachen**

Zeigen Sie, dass folgende Sprache L nicht kontextfrei ist:

$$L = \left\{0^l 10^l 10^l \mid l \in \mathbb{N}_0\right\}$$

Lösung:

Angenommen L sei kontextfrei, dann besagt das Pumping-Lemma für kontextfreie Sprachen, dass eine Zahl $k \in \mathbb{N}$ existiert, sodass sich jedes Wort $z \in L$ mit $|z| \geq k$ zerlegen lässt zu $z = uvwxy$ mit

1) $|vwx| \leq k$,

2) $vx \neq \lambda$,

3) $\forall i \in N_0$ gilt $uv^i wx^i y \in L$.

Diese Folgerung führen wir zum Widerspruch. Wir wählen dazu das Wort $z = 0^k 10^k 10^k \in L$ mit $|z| \geq k$ und zeigen, dass für jede mögliche Zerlegung von z, die 1) und 2) erfüllt, 3) nicht erfüllt sein kann. (Man beachte, dass in diesem Fall die Struktur des Pumpwortes z alle Wörter aus L umfasst – in dem Sinne, dass ein durch alle natürlichen Zahlen durchlaufendes k alle Wörter aus L ergeben würde.)
Wegen 1) kann vwx nicht beide Einsen von z enthalten, und auch nur Nullen aus maximal zwei der drei durch die Einsen abgegrenzten Bereiche:

a) Bereich vor der ersten Eins,

b) Bereich zwischen der ersten und zweiten Eins,

c) Bereich nach der zweiten Eins.

Es ergeben sich zwei mögliche Fälle, die betrachtet werden müssen, um zum Widerspruch zu kommen:

- ***vwx* enthält keine Eins**: dann muss vwx mindestens eine 0 enthalten, da $vx \neq \lambda$. Da der Wortteil vx nicht gleichzeitig Nullen aus allen drei Bereichen enthalten kann (wegen $|vwx| \leq k$), kann mit beliebigem $i \neq 1$ gepumpt werden, um zu einem Widerspruch zu kommen: Das gepumpte Wort sei $z' =_{def} uv^i wx^i y$ mit $i \neq 1$. Die Pumpstellen v und x enthalten Nullen aus mindestens einem der drei Bereiche (Bereich X) und aus mindestens einem nicht (Bereich Y). Durch das Pumpen ändert sich die Anzahl der Nullen in X (sie wird um mindestens eine Null kleiner, falls wir $i = 0$ wählen und um mindestens eine Null größer, falls wir $i > 1$ wählen), aber nicht die Anzahl der Nullen in Y. Also stimmt die Anzahl der Nullen in den drei Bereichen nicht mehr überein, was nicht der Definition der Sprache entspricht. Es gilt $z' \notin L$.

- ***vwx* enthält genau eine Eins**: Dieser Fall unterteilt sich in zwei weitere Fälle:
  - **Die Eins ist nicht in *vx* enthalten** (sondern in w): dann enthält vx wieder Nullen aus höchstens zwei der drei Bereiche und es gilt mit derselben Argumentation wie im ersten Fall, dass $z' =_{def} uv^i wx^i y$ für alle $i \neq 1$ nicht in L liegen kann, also $z' \notin L$.
  - **Die Eins ist in *vx* enthalten**: dann enthält vx entweder keine Nullen oder Nullen aus höchstens zwei der drei Bereiche; das ist zum Pumpen ungeschickt, denn wir wissen nicht, ob sich die Anzahl der Nullen dabei ändern würde oder nicht. Wir argumentieren daher einfacher über die Anzahl der Einsen: wenn mit irgendeinem $i \neq 1$ gepumpt wird, dann verändert sich die Anzahl der Einsen von vwx zu $v^i wx^i$ (sie wird um mindestens eine Eins kleiner für $i = 0$ und um mindestens eine Eins größer für $i > 1$) und damit vom ungepumpten Wort z zum gepumpten Wort $z' =_{def} uv^i wx^i y$. Da laut Sprachdefinition die Anzahl der Einsen immer zwei sein muss, gilt wieder $z' \notin L$.

Für alle Fälle existiert also ein i (sogar unendlich viele verschiedene i, da jeweils mit allen $i \neq 1$ gepumpt werden kann), sodass das gepumpte Wort $uv^i wx^i y \notin L$. Damit ergibt sich ein Widerspruch, da laut Pumping-Lemma alle gepumpten Wörter ebenfalls in L liegen müssten. Daher müssen wir die Annahme aufgeben, L sei kontextfrei.

Aufgabe 66 ⋆⋆

PUM-AG **Pumping-Lemma für kontextfreie Sprachen**

Zeigen Sie, dass folgende Sprache L nicht kontextfrei ist:

$$L = \{0^i 1^j 2^i 3^j \mid i, j \in \mathbb{N}\}$$

Lösung:

Angenommen L sei kontextfrei, dann existiert laut Pumping-Lemma für kontextfreie Sprachen eine Zahl $k \in \mathbb{N}$, sodass sich jedes Wort $z \in L$ mit $|z| \geq k$ zerlegen lässt als $z = uvwxy$ mit

1) $|vwx| \leq k$
2) $|vx| \geq 1$
3) $\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$

Um zum Widerspruch zu kommen, wählen wir $z = 0^k 1^k 2^k 3^k \in L$ mit $|z| \geq k$. Da $|vwx| \leq k$ und $|vx| \geq 1$, besteht vwx (und damit insbesondere auch vx) entweder aus einer nichtleeren Kette eines einzigen Zeichens (0, 1, 2 oder 3) oder aus zwei verschiedenen Zeichen (0 und 1, 1 und 2 oder 2 und 3). Es ergeben sich drei wesentliche Fälle für die Zeichen in vx, wenn das Vorkommen nur eines Zeichens als Spezialfall des Vorkommens von zwei Zeichen (mit einem Zeichen null Mal vorkommend) betrachtet wird. Seien $s, t \in \mathbb{N}_0$, die Pumpvariable $i = 0$ und damit das gepumpte Wort $z' =_{def} uv^0 wx^0 y = uwy$. Die drei Fälle sind:

- ***vx* enthält keine 2 und keine 3**, also $vx = 0^s 1^t$: dann ist $z' = uwy = 0^{k-s} 1^{k-t} 2^k 3^k$.
- ***vx* enthält keine 0 und keine 3**, also $vx = 1^s 2^t$: dann ist $z' = uwy = 0^k 1^{k-s} 2^{k-t} 3^k$.
- ***vx* enthält keine 0 und keine 1**, also $vx = 2^s 3^t$: dann ist $z' = uwy = 0^k 1^k 2^{k-s} 3^{k-t}$.

In jedem Fall gilt $k \neq k - s$ oder $k \neq k - t$, wegen $|vx| = s + t \geq 1$. Dadurch ist in z'

- **im ersten Fall** entweder die Anzahl der Nullen kleiner als die Anzahl der Zweien oder die Anzahl der Einsen kleiner als die Anzahl der Dreien;
- **im zweiten Fall** entweder die Anzahl der Nullen größer als die Anzahl der Zweien oder die Anzahl der Einsen kleiner als die Anzahl der Dreien;
- **im dritten Fall** entweder die Anzahl der Nullen größer als die Anzahl der Zweien oder die Anzahl der Einsen größer als die Anzahl der Dreien.

Da die jeweiligen Anzahlen laut Sprachdefinition gleich sein müssten, kann z' nicht in der Sprache liegen und es gilt $z' \notin L$. Damit ergibt sich ein Widerspruch zu 3) und wir müssen die Annahme, L sei kontextfrei, aufgeben.
Bemerkung: Es mag überraschen, dass eine Sprache, die eine so einfache Struktur hat wie $L = \{0^i 1^j 2^i 3^j \mid i, j \in \mathbb{N}\}$ nicht kontextfrei ist, während "ähnliche" Sprachen wie beispielsweise $L' = \{0^i 1^i 2^j 3^j \mid i, j \in \mathbb{N}\}$ oder $L'' = \{0^i 1^j 2^j 3^i \mid i, j \in \mathbb{N}\}$ kontextfrei sind. Der Grund für die Schwierigkeit von L liegt in der Verschränkung der Blöcke, die jeweils gleich viele Zeichen haben sollen, ineinander. Salopp gesagt müsste ein Kellerautomat sich zunächst sowohl die Anzahl der Nullen als auch die Anzahl der Einsen merken, indem er entsprechend viele Zeichen in den Keller schreibt; dann müsste er die Anzahl der Nullen mit der Anzahl der Zweien vergleichen, die Nullen liegen aber unter den Einsen und können wegen des LIFO-Prinzips nicht betrachtet werden.

Aufgabe 67 ⋆⋆

PUM-AI **Pumping-Lemma für kontextfreie Sprachen**

Gegeben sei die Sprache L mit

$$L = \{(ab)^n(ca)^n(bc)^n \mid n \in \mathbb{N}_0\}$$

Gibt es eine kontextfreie Grammatik, die L erzeugt? Wenn ja, geben Sie eine solche Grammatik an, wenn nein, begründen Sie, warum es keine gibt.

Lösung:

Die Sprache L ist nicht kontextfrei – was wegen der Zuordnung der Aufgabe zum Pumping-Lemma-Kapitel vielleicht nicht sehr überraschend ist. Wir zeigen durch einen Widerspruchsbeweis, dass es keine kontextfreie Grammatik für L geben kann.
Angenommen, es gäbe eine solche Grammatik für L, dann existiert laut Pumping-Lemma für kontextfreie Sprachen eine Zahl $k \in \mathbb{N}$, sodass sich jedes Wort $z \in L$ mit $|z| \geq k$ zerlegen lässt als $z = uvwxy$ mit

1) $|vwx| \leq k$,
2) $|vx| \geq 1$,
3) $\forall i \in \mathbb{N}_0 : uv^iwx^iy \in L$.

Um diese Folgerung zum Widerspruch zu führen, sei $z = (ab)^k(ca)^k(bc)^k \in L$ mit $|z| \geq k$. Sei weiter die Pumpvariable $i = 0$, es gilt also für das gepumpte Wort $z' =_{def} uv^0wx^0y = uwy$. Wir unterscheiden zwei Fälle je nach Struktur der Zerlegung in u, v, w, x und y. Dabei interessiert besonders die Belegung der Pumpstellen x und y.

- ***v* und *x* enthalten nur Teilausdrücke, die "wohlgeformt" sind** (also $(ab)^\star$, $(ca)^\star$, $(bc)^\star$ aber beispielsweise nicht $(ab)^\star a$, $(ca)^\star c$ oder $(bc)^\star b$): dann betrachten wir zwei Fälle.
  - Liegt vwx im vorderen Teil von z, enthält vwx also (ab), (ca) oder beides, aber keines der hinteren (bc)'s (wegen $|vwx| \leq k$ kann es nicht alle drei enthalten), dann enthält $z' = uwy$ zu wenige (ab)'s oder zu wenige (ca)'s im Vergleich zur Anzahl der (bc)'s und demnach gilt $z' \notin L$.
  - Liegt vwx im hinteren Teil von z, enthält vwx also (ca), (bc) oder beides, aber keines der vorderen (ab)'s, dann enthält $z' = uwy$ zu wenige (ca)'s oder zu wenige (bc)'s im Vergleich zur Anzahl der (ab)'s und demnach gilt $z' \notin L$.
- ***v* oder *x* enthält Teilausdrücke, die nicht "wohlgeformt" sind**: dann wird beim Pumpen die Struktur der wohlgeformten Ausdrücke verletzt, sodass $z' \notin L$ gilt. (Das sieht man einfacher, wenn $i = 2$ gewählt wird; dann kommt nach dem Pumpen obiger Beispiele für nicht-wohlgeformte Ausdrücke jeweils ein Zeichen zweimal hintereinander vor, was nicht erlaubt ist. Strenggenommen müssten hier eigentlich alle möglichen Fälle betrachtet werden, um den Beweis abzuschließen.)

Damit ergibt sich in jedem der Fälle ein Widerspruch zu 3). Man muss daher die Annahme aufgeben, dass eine kontextfreie Grammatik existiert, die L erzeugt, und L ist nicht kontextfrei.

Aufgabe 68 ★⋆

PUM-AJ **Pumping-Lemma für kontextfreie Sprachen**

Gegeben sei die Sprache L mit

$$L = \{AB \in \{0,1\}^\star \mid |A|_0 = |B|_0 \text{ und } |A|_1 = |B|_1\}$$

Zeigen Sie, dass L nicht kontextfrei ist.

Lösung:

Angenommen, L sei kontextfrei, dann besagt das Pumping-Lemma für kontextfreie Sprachen, dass es ein $k \in \mathbb{N}$ gibt, sodass sich jedes Wort $z \in L$ mit $|z| \geq k$ zerlegen lässt in $z = uvwxy$ mit

1) $|vwx| \leq k$,
2) $vx \neq \lambda$ (diese alternative Schreibweise bedeutet dasselbe wie $|vx| > 0$),
3) $\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$.

Um einen Widerspruch herbeizuführen, betrachten wir das Wort $z = 0^k 1^k 0^k 1^k \in L$ mit $|z| \geq k$. Sei $z = AB$ mit $|A| = |B|$. Wir bezeichnen A als die vordere Hälfte des Wortes z und B als die hintere Hälfte. Im Folgenden unterscheiden wir zwei Fälle:

- ***vx* enthält unterschiedlich viele Zeichen aus *A* und *B***: dann führt ein Pumpen mit $i = 0$ dazu, dass sich die Mitte des Wortes nach links oder rechts verschiebt. Dabei werden aus der einen Hälfte immer mehr Zeichen von einer Sorte gelöscht als aus der anderen, da wegen 1) vx nicht Einsen oder Nullen aus beiden Hälften enthalten kann. Das bedeutet, dass die Anzahl der Einsen bzw. Nullen in der hinteren bzw. vorderen Hälfte zunimmt und in der vorderen bzw. hinteren abnimmt. Da laut Definition von L die Anzahl der Nullen und Einsen in der vorderen und hinteren Hälfte gleich sein muss, ist für diesen Fall $z' =_{def} uv^0wx^0y \notin L$.
- ***vx* enthält gleich viele Zeichen aus *A* und *B***: Hier führt ein Pumpen mit $i = 0$ nicht dazu, dass sich die Mitte des Wortes verschiebt. Da vx wegen 1) und 2) jedoch von der Form $vx = 1^p 0^p$ mit $p \in \mathbb{N}$ ist, bedeutet dies, dass die Anzahl der Einsen in der vorderen Hälfte und die Anzahl der Nullen in der hinteren Hälfte zunimmt, während die Anzahl der Einsen in der hinteren und die Anzahl der Nullen in der vorderen gleich bleibt. Dies ist erneut ein Widerspruch zur Definition von L, also gilt auch für diesen Fall $z' \notin L$.

Also gibt es für das Wort z keine Zerlegung mit den drei Pumpeigenschaften. Da es laut Pumping-Lemma für jedes Wort der Sprache L mit mindestens der Länge k eine solche Zerlegung geben müsste, müssen wir unsere ursprüngliche Annahme, dass L kontextfrei sei, aufgeben.

Aufgabe 69 ★⋆

PUM-AK **Pumping-Lemma für kontextfreie Sprachen**

Gegeben sei die Sprache L mit

$$L = \{a^n b^m c^{n \cdot m} \mid n, m \in \mathbb{N}\}$$

Zeigen Sie, dass L nicht kontextfrei ist.

Lösung:

Angenommen L sei kontextfrei, dann besagt das Pumping-Lemma für kontextfreie Sprachen, dass es ein $k \in \mathbb{N}$ gibt, so dass sich jedes Wort $z \in L$ mit $|z| \geq k$ zerlegen lässt in $z = uvwxy$ mit:

1) $|vwx| \leq k$,
2) $|vx| > 0$,
3) $\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$.

Um einen Widerspruch herbeizuführen und die Annahme als falsch zu beweisen, betrachten wir das Wort $z = a^k b^k c^{k^2} \in L$ mit $|z| = 4k \geq k$. Da $|vwx| \leq k$, kann vx nicht a, b und c gleichzeitig enthalten. Sei $i = 2$ und das gepumpte Wort $z' =_{def} uv^2wx^2y$. Wir unterscheiden im Folgenden zwei Fälle:

- ***vx* enthält kein *c***: dann enthält vx mindestens ein a oder ein b. Da wegen $|vx| > 0$ die Pumpvariablen v und x nicht beide leer sein dürfen, muss ein Pumpen mit $i = 2$ dazu führen, dass das Wort z' mehr a's und/oder mehr b's als z enthält. Da die Anzahl der c's für ein Wort in L das Produkt der Anzahl der a's und b's ist, müsste auch die Zahl der c's zunehmen, damit $z' \in L$ gilt. Das Teilwort vx enthält aber kein c, es gilt also $z' \notin L$.
- ***vx* enthält kein *a***: dann enthält vx mindestens ein b oder ein c. Hier führt ein Pumpen mit $i = 2$ dazu, dass das Wort z' mehr b's und/oder mehr c's enthält, als z. Da die Anzahl der c's für ein Wort in L aber das Produkt der Anzahl der a's und b's ist und da z genau k-mal a enthält, müssten für jedes gepumpte b k zusätzliche c's entstehen. Das bedeutet, dass vx entweder mindestens $k + 1$ Zeichen enthalten müsste (ein b und k c's), was 1) widersprechen würde, oder leer sein müsste, was 2) widersprechen würde. Es gilt also auch hier $z' \notin L$.

Es gibt also keine Zerlegung des Wortes z, für die alle Bedingungen des Pumping-Lemmas gelten. Unsere Annahme, dass L kontextfrei ist, muss falsch sein.

Aufgabe 70 ⋆⋆

PUM-AM **Pumping-Lemma für kontextfreie Sprachen**

Ist die folgende Sprache L kontextfrei?

$$L = \{a^i b^j c^i \mid i \geq j \in \mathbb{N}\}$$

Lösung:

Angenommen, L sei kontextfrei, dann existiert eine Konstante $k \in \mathbb{N}$, sodass für alle Wörter $z \in L$ mit $|z| \geq k$ Wörter $u, v, w, x, y \in \{a, b, c\}^\star$ existieren mit $z = uvwxy$, sodass gilt:

1) $|vwx| \leq k$,
2) $|vx| \geq 1$,
3) $\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$.

Um zum Widerspruch zu kommen, setzen wir $z = a^k b^k c^k \in L$ mit $|z| = 3k \geq k$. Aus $|vwx| \leq k$ folgt, dass vx maximal zwei der drei Symbole a, b und c enthalten kann. Wir unterscheiden folgende Fälle:

- ***vx* enthält mindestens ein *a***: dann enthält es kein c:

  $$|vx|_a > 0 \Rightarrow |vx|_c = 0$$

  Jedes $z' =_{def} uv^i wx^i y$ mit $i \neq 1$ ist dann nicht Element der Sprache, weil $|z'|_a \neq |z'|_c$. Es gilt also $z' \notin L$.

- ***vx* enthält kein *a***: dann enthält es mindestens ein b oder ein c:

  $$|vx|_a = 0 \Rightarrow |vx|_b > 0 \text{ oder } |vx|_c > 0$$

  Es folgt $z' = uv^i wx^i y \notin L$ für alle $i > 1$, weil dann $|z'|_a < |z'|_b$ oder $|z'|_a \neq |z'|_c$ (oder beides) gilt.

Daraus folgt ein Widerspruch, denn laut Pumping-Lemma sollte jedes gepumpte Wort ebenfalls in L liegen. Also müssen wir die ursprüngliche Annahme aufgeben, dass L eine kontextfreie Sprache ist.

Aufgabe 71 ★★⋆

PUM-AH **Pumping-Lemma für kontextfreie Sprachen**

Gegeben sei die Sprache L mit

$$L = \{a^k b^{k^2} \mid k \geq 0\}$$

Zeigen Sie, dass L nicht kontextfrei ist.

Lösung:

Angenommen L sei kontextfrei, dann existiert laut Pumping-Lemma für kontextfreie Sprachen eine Zahl $k \in \mathbb{N}$, sodass sich jedes Wort $z \in L$ mit $|z| \geq k$ zerlegen lässt zu $z = uvwxy$ mit

1) $|vwx| \leq k$,

2) $|vx| \geq 1$,

3) $\forall i \in \mathbb{N}_0 : uv^i wx^i y \in L$.

Wir zeigen durch ein Gegenbeispiel, dass so eine Zerlegung nicht immer existiert, und führen damit die Annahme zum Widerspruch. Dazu wählen wir $z = a^k b^{k^2} \in L$ (mit $|z| \geq k$) und $i = 2$, sodass sich das gepumpte Wort zu $z' =_{def} uv^2wx^2y$ ergibt. Das Teilwort vwx bzw. der zu pumpende Teil vx kann nur a's, nur b's oder sowohl a's als auch b's enthalten; vx muss außerdem mindestens ein Zeichen enthalten, da $|vx| \geq 1$. Wir unterscheiden die folgenden Fälle:

- ***vx* enthält nur *a*'s oder nur *b*'s**: dann enthält $z' = uv^2wx^2y$ zu viele a's beziehungsweise zu viele b's, denn nur eines der beiden Zeichen wurde gepumpt. Damit gilt auf jeden Fall $|z'|_a^2 \neq |z'|_b$, und die Definition der Sprache ist nicht erfüllt; es gilt $z' \notin L$.

- ***vx* enthält sowohl *a*'s als auch *b*'s**: dann ist $vx = a^l b^m$ für $l, m \in \mathbb{N}_0$ mit $l + m \leq k$, denn es gilt $|vwx| \leq k$. Nun hängt der weitere Beweis von der konkreten Belegung von v und x ab. Wir unterscheiden folgende Fälle:
  - ***v* enthält nur *a*'s und *x* enthält nur *b*'s**: dann ist $v = a^l$ und $x = b^m$. Damit gilt für die Anzahl der a's bzw. b's im gepumpten Wort $z' = uv^2wx^2y$:

    $$|z'|_a = |z|_a + l = k + l, \; |z'|_b = |z|_b + m = k^2 + m$$

    bzw. für das gepumpte Wort z':

    $$z' = a^{k+l} b^{k^2+m} \stackrel{?}{=} a^q b^{q^2}$$

    Nun stellt sich die Frage, ob es ein $q \in \mathbb{N}$ gibt, sodass $q = k + l$ und $q^2 = k^2 + m$, denn wenn es ein solches q nicht geben sollte, wäre $z' \notin L$, da die Anzahl der b's nicht das Quadrat der Anzahl der a's wäre. Wenn es so ein q geben sollte, müsste gelten:

    $$(k + l)^2 = k^2 + m \Leftrightarrow l(l + 2k) = m$$

Eine Lösung der Gleichung, die $l = 0$ oder $m = 0$ beinhaltet, können wir ausschließen, denn das widerspricht der Forderung "vx enthält sowohl a's als auch b's". Sei also $l, m \geq 1$. Wegen $k \geq |vwx| \geq m$ muss gelten: $l(l + 2m) \leq m$ (Abschätzung, indem k auf der linken Seite der Gleichung durch m ersetzt wird). Offenbar ist aber für $l, m \geq 1$

$$l(l + 2m) = l^2 + 2lm > m$$

Das ist ein Widerspruch; damit konnten wir zeigen, dass es kein q wie oben gefordert geben kann und dass somit $z' \notin L$.

– **Entweder v oder x enthält sowohl a's als auch b's**: dann enthält entweder v^2 oder x^2 das Teilwort ba, denn "$(a \ldots ab \ldots b)^2 = a \ldots ab \ldots \underline{ba} \ldots ab \ldots b$"; damit enthält auch $z' = uv^2wx^2y$ dieses Teilwort, was nicht der Sprachdefinition von L entspricht, und es gilt $z' \notin L$.

Damit ergibt sich in jedem der insgesamt drei Fälle ein Widerspruch zu der Bedingung 3). Wir müssen (und dürfen) daher die Annahme aufgeben, dass eine kontextfreie Grammatik existiert, die L erzeugt. L ist also nicht kontextfrei.

Aufgabe 72 ★⋆

PUM-AE **Pumping-Lemma für kontextfreie Sprachen**

Entscheiden Sie, ob die folgenden Aussagen wahr oder falsch sind.

Lösung:

	Wahr	Falsch
Es gibt nicht-kontextfreie Sprachen, die das Pumping-Lemma für kontextfreie Sprachen erfüllen.	☒	☐

Erklärung: Die beiden Pumping-Lemmata sind Eigenschaften aller regulären bzw. kontextfreien Sprachen, allerdings nicht ausschließlich, denn es gibt auch nicht-reguläre bzw. nicht-kontextfreie Sprachen, die das jeweilige Pumping-Lemma erfüllen.

Man bedient sich des Pumping-Lemmas für kontextfreie Sprachen, um in einem Widerspruchsbeweis zu zeigen, dass eine Sprache nicht kontextfrei ist.	☒	☐

Erklärung: Beide Pumping-Lemmata werden meist in dieser Form verwendet.

Die Konstante k des Pumping-Lemmas für kontextfreie Sprachen ergibt sich aus der Anzahl der Zustände eines endlichen Automaten.	☐	☒

Erklärung: Die Konstante n des Pumping-Lemmas für reguläre Sprachen ergibt sich aus der Anzahl der Zustände eines endlichen Automaten, die Konstante k des Pumping-Lemmas für kontextfreie Sprachen aus der Anzahl der Nonterminalsymbole einer Grammatik (allerdings ist k nicht genau die Anzahl dieser Nonterminalsymbole, sondern hängt exponentiell davon ab).

Mit dem Algorithmus von Cocke, Younger und Kasami lässt sich für jede nicht-kontextfreie Sprache ein Pumpwort für einen Widerspruchsbeweis mit dem Pumping-Lemma konstruieren.	☐	☒

Erklärung: Der Algorithmus von Cocke, Younger und Kasami überprüft, ob ein Wort zu einer kontextfreien Sprache gehört, er erzeugt jedoch (leider) keine Pumpwörter für das Pumping-Lemma.

Das Pumping-Lemma für kontextfreie Sprachen hat zwei, das für reguläre Sprachen nur eine Pumpstelle.	☒	☐

Erklärung: Die eine Pumpstelle beim Pumping-Lemma für reguläre Sprachen ergibt sich aus einer notwendigen Schleife in einem endlichen Automaten, die zwei bei dem für kontextfreie Sprachen aus dem Ableitungsbaum einer kontextfreien Grammatik.

Für einen Widerspruchsbeweis mit dem Pumping-Lemma für kontextfreie Sprachen kann man dieses Pumping-Lemma immer durch das Pumping-Lemma für reguläre Sprachen ersetzen, indem man für eine der beiden Pumpvariablen das leere Wort wählt.	☐	☒

Erklärung: Abgesehen davon, dass die Aussage insgesamt nicht viel Sinn ergibt, kann insbesondere in einem Widerspruchsbeweis mit einem Pumping-Lemma die Zerlegung nicht gewählt werden, sondern es muss über alle möglichen Belegungen argumentiert werden.

8 Turingmaschinen

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=353

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 73 ⋆⋆

TUR-AA **Turingmaschinen**

Geben Sie eine Turingmaschine A an, die die als Binärzahl interpretierte Bandinschrift $w \in \{0,1\}^\star$ in die Zweikomplement-Darstellung w' umwandelt (vgl. Band 2, Kapitel 5). Geben Sie A vollständig an.
Hinweise:

- Der Schreib-/Lesekopf steht zu Beginn über dem linkesten Zeichen des Eingabewortes w.
- A soll über dem linkesten Zeichen des konvertierten Wortes w' halten.

Lösung:

Das Zweikomplement wird gebildet durch Kippen aller Bits und Addition von 1.
Vorgehensweise:

- w endet auf 1: Schreibe eine 1 und invertiere alle Zeichen links davon.
- w endet auf 0: Schreibe eine 0 und suche von rechts aus die erste 1. Invertiere alle Zeichen links von dieser 1.

$$A = (E, B, S, \delta, s_0, F) \text{ mit } E = \{0,1\}, B = \{0,1,\star\}, S = \{s_0, s_1, s_2, s_e\}, F = \{s_e\}$$

δ	0	1	$\star$
s_0	$(s_0, 0, R)$	$(s_0, 1, R)$	$(s_1, \star, L)$
s_1	$(s_1, 0, L)$	$(s_2, 1, L)$	$(s_e, \star, R)$
s_2	$(s_2, 1, L)$	$(s_2, 0, L)$	$(s_e, \star, R)$
s_e			

Aufgabe 74 ⋆⋆

TUR-AF **Turingmaschinen**

Entwerfen Sie eine Turingmaschine A, welche eine gegebene Binärfolge $w \in \{0,1\}^\star$ um ihr Komplement erweitert, beispielsweise wird aus der Folge 111 die Folge 111000, aus 011 wird die Folge 011100, und aus 001 wird 001110. Geben Sie A vollständig an.

Lösung:

Vorgehensweise:

- A erstellt das Komplement auf dem Band im Anschluss an die Folge aus Binärzeichen codiert mit den Zeichen E für 1 und N für 0.
- Sie ersetzt jedes bereits verneinte Zeichen durch ein E für 1 bzw. N für 0.
- Wenn das komplette Band nur noch aus E und N besteht, ersetzt A diese durch die entsprechenden Binärzeichen 0 und 1.

$A = (E, B, S, \delta, s_0, F); E = \{0, 1\}, B = \{0, 1, E, N, \star\}, S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_e\}, F = \{s_e\}$

δ	1	0	$\star$	E	N
s_0	(s_1, E, R)	(s_4, N, R)	$(s_e, \star, N)$		
s_1	$(s_1, 1, R)$	$(s_1, 0, R)$	(s_2, N, L)	(s_1, E, R)	(s_1, N, R)
s_2	$(s_3, 1, L)$	$(s_3, 0, L)$	$(s_5, \star, R)$	(s_2, E, L)	(s_2, N, L)
s_3	$(s_3, 1, L)$	$(s_3, 0, L)$		(s_0, E, R)	(s_0, N, R)
s_4	$(s_4, 1, R)$	$(s_4, 0, R)$	(s_2, E, L)	(s_4, E, R)	(s_4, N, R)
s_5			$(s_6, \star, L)$	$(s_5, 1, R)$	$(s_5, 0, R)$
s_6	$(s_6, 1, L)$	$(s_6, 0, L)$	$(s_e, \star, R)$		
s_e					

Aufgabe 75 ★⋆

TUR-AH **Turingmaschinen**

Geben Sie eine Turingmaschine A an, die die Bandinschrift $w \in \{a, b, c\}^\star$ um ein Feld nach links verschiebt. Definieren Sie A vollständig.

Lösung:

Vorgehensweise:

- s_0: Gehe auf äußerst rechtes Zeichen.
- s_1: Äußerst rechtes Zeichen merken und löschen.
- s_a: Gemerktes a schreiben und aktuelles Zeichen merken.
- s_b: Gemerktes b schreiben und aktuelles Zeichen merken.
- s_c: Gemerktes c schreiben und aktuelles Zeichen merken.
- s_e: Endzustand.

$A = (E, B, S, \delta, s_0, F)$ mit $E = \{a, b, c\}, S = \{s_0, s_1, s_a, s_b, s_c, s_e\}, B = \{a, b, c, \star\}, F = \{s_e\}$

δ	a	b	c	$\star$
s_0	(s_0, a, R)	(s_0, b, R)	(s_0, c, R)	$(s_1, \star, L)$
s_1	$(s_a, \star, L)$	$(s_b, \star, L)$	$(s_c, \star, L)$	$(s_e, \star, N)$
s_a	(s_a, a, L)	(s_b, a, L)	(s_c, a, L)	(s_e, a, N)
s_b	(s_a, b, L)	(s_b, b, L)	(s_c, b, L)	(s_e, b, N)
s_c	(s_a, c, L)	(s_b, c, L)	(s_c, c, L)	(s_e, c, N)
s_e				

Anmerkung:
Diese Aufgabe ist zwar eine gute Übung, so eine Turingmaschine hätte aber keinen praktischen Nutzen, da sich aus Sicht der Turingmaschine nichts ändert, wenn die Eingabe auf dem unendlichen Band irgendwohin verschoben sind.

Aufgabe 76 $\star^\star$

TUR-AC **Turingmaschinen**

Konstruieren Sie eine Turingmaschine A, die rechts neben der Eingabe $w \in \{0, 1\}^\star$ nochmals das Wort w schreibt. Geben Sie A vollständig an.

Lösung:

Vorgehensweise:

- Gestartet wird links am Wort w.
- Wenn eine 1 gelesen wird, wird e geschrieben und ganz nach rechts gegangen, um auch dort ein e schreiben.
- Wenn eine 0 gelesen wird, wird n geschrieben und ganz nach rechts gegangen, um dort ein n zu schreiben.
- Wieder nach links gehen und bei jeder gelesenen 1 oder 0 den Vorgang wiederholen.
- Am Ende wird aus jedem e eine 1 und aus jedem n eine 0 gemacht.

$A = (E, B, S, \delta, s_0, F)$ mit $E = \{0, 1\}, B = \{0, 1, n, e, \star\}, S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_e\}, F = \{s_e\}$

δ	0	1	$\star$	n	e
s_0	(s_1, n, R)	(s_4, e, R)	$(s_5, \star, L)$	$(s_0, 0, R)$	$(s_0, 1, R)$
s_1	$(s_1, 0, R)$	$(s_1, 1, R)$	(s_2, n, L)	(s_1, n, R)	(s_1, e, R)
s_2	$(s_3, 0, L)$	$(s_3, 1, L)$	$(s_0, \star, R)$	(s_2, n, L)	(s_2, e, L)
s_3	$(s_3, 0, L)$	$(s_3, 1, L)$		(s_0, n, R)	(s_0, e, R)
s_4	$(s_4, 0, R)$	$(s_4, 1, R)$	(s_2, e, L)	(s_4, n, R)	(s_4, e, R)
s_5	$(s_5, 0, L)$	$(s_5, 1, L)$	$(s_e, \star, R)$		
s_e					

Aufgabe 77 ★★

TUR-AG **Turingmaschinen**

Gegeben sei die Sprache

$$L = \{w \in \{a, b, c\}^+ \mid w = w'\}$$

L ist also eine Sprache, die nur Palindrome enthält. Ein Palindrom ist ein Wort, das von hinten nach vorne gelesen das gleiche ergibt wie von vorne nach hinten (beispielsweise “abba” oder über anderen Alphabeten “otto”, “reittier”). Einelementige Wörter sind Palindrome, die in L enthalten sind. Das leere Wort ist ein Palindrom, das nicht in L enthalten ist.

a) Geben Sie eine kontextfreie Grammatik G an, für die gilt $L(G) = L$.

Lösung:

Die kontextfreie Grammatik lautet

$$\begin{aligned} G &= (N, T, P, S) \\ N &= \{S, D\} \\ T &= \{a, b, c\} \\ P &= \{S \rightarrow aDa \mid bDb \mid cDc \mid a \mid b \mid c, \\ &\quad\; D \rightarrow aDa \mid bDb \mid cDc \mid a \mid b \mid c \mid \lambda\} \end{aligned}$$

b) Geben Sie eine Turingmaschine A an, die Wörter der Sprache L akzeptiert, für die also gilt $L(A) = L(G) = L$. Definieren Sie A vollständig.

Lösung:

Vorgehensweise:

- Überschreibt erstes Zeichen mit ⋆ und merkt sich das Zeichen im Zustand s_a, s_b oder s_c; läuft bis zum Bandende, wechselt entsprechend in s_{aa}, s_{bb} oder s_{cc}
- 1) falls letztes Zeichen gleich überschreibt dieses Zeichen mit ⋆, läuft mit s_L zurück bis zu ⋆ und Neuanfang in s_{00}
  2) falls letztes Zeichen ungleich und nicht ⋆ bleibt A stehen
  3) falls letztes Zeichen ⋆ (Wort mit ungerader Länge) geht A in den Endzustand s_e

$$A = (E, B, S, \delta, s_0, F) \text{ mit } E = \{a, b, c\}, B = \{a, b, c, \star\},$$

$$S = \{s_0, s_{00}, s_a, s_{aa}, s_b, s_{bb}, s_c, s_{cc}, s_L, s_e\}, F = \{s_e\}$$

δ	a	b	c	$\star$
s_0	$(s_a, \star, R)$	$(s_b, \star, R)$	$(s_c, \star, R)$	
s_{00}	$(s_a, \star, R)$	$(s_b, \star, R)$	$(s_c, \star, R)$	$(s_e, \star, N)$
s_a	(s_a, a, R)	(s_a, b, R)	(s_a, c, R)	$(s_{aa}, \star, L)$
s_{aa}	$(s_L, \star, L)$			$(s_e, \star, N)$
s_b	(s_b, a, R)	(s_b, b, R)	(s_b, c, R)	$(s_{bb}, \star, L)$
s_{bb}		$(s_L, \star, L)$		$(s_e, \star, N)$
s_c	(s_c, a, R)	(s_c, b, R)	(s_c, c, R)	$(s_{cc}, \star, L)$
s_{cc}			$(s_L, \star, L)$	$(s_e, \star, N)$
s_L	(s_L, a, L)	(s_L, b, L)	(s_L, c, L)	$(s_{00}, \star, R)$
s_e				

Aufgabe 78 ⋆

TUR-AE **Turingmaschinen**

Definieren Sie eine Turingmaschine A, welche bei einer Eingabe $w \in \{0,1\}^\star$ folgende Funktion $f : \{0,1\}^\star \to \{0,1\}^\star$ berechnet.

$$f(w) \to \begin{cases} w1, & \text{falls } |w|_1 = 2v + 1 \text{ mit } v \in \mathbb{N}_0 \\ w0 & \text{sonst} \end{cases}$$

Die Turingmaschine A erkennt also, ob eine gerade oder ungerade Anzahl an Einsen vorkommt. Bei einer ungeraden Anzahl schreibt sie rechts neben die Bandeinschrift eine 1, bei gerader Anzahl eine 0. Geben Sie A vollständig an.

Lösung:

Vorgehensweise:

- s_0: gerade Anzahl an Einsen; schreibt bei $\star$ eine 0
- s_1: ungerade Anzahl an Einsen; schreibt bei $\star$ eine 1

$A = (E, B, S, \delta, s_0, F)$ mit $E = \{0,1\}, B = \{0,1,\star\}, S = \{s_0, s_1, s_2, s_e\}, F = \{s_e\}$

δ	0	1	$\star$
s_0	$(s_0, 0, R)$	$(s_1, 1, R)$	$(s_2, 0, L)$
s_1	$(s_1, 0, R)$	$(s_0, 1, R)$	$(s_2, 1, L)$
s_2	$(s_2, 0, L)$	$(s_2, 1, L)$	$(s_e, *, R)$
s_e			

Aufgabe 79 $\star^{\star^{\star}}$

TUR-AD **Turingmaschinen**

Geben Sie eine Turingmaschine A bzw. A' an, die für eine Eingabe $w \in \{0,1\}^\star$ jeweils folgende Funktionen $f : \{0,1\}^\star \to \{0,1\}^\star$ bzw. $f' : \{0,1\}^\star \to \{0,1\}^\star$ berechnet.

a)

$$f(w) \to \begin{cases} wu, & \text{falls } w = uv \text{ für } u, v \in \{0,1\}^\star \text{ und } |u| = |v| \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Die Turingmaschine überprüft also, ob $w = uv$ für $u, v \in \{0,1\}^\star$ und u hat die gleiche Wortlänge wie v. In diesem Fall schreibt sie das Wort u noch einmal hinter die Eingabe. Falls die Eingabe nicht diese Form hat, ist das Verhalten von A undefiniert. In diesem Fall soll A lediglich in einem Nicht-Endzustand halten. Geben Sie A vollständig an.

Lösung:

Vorgehensweise:

- Links neben der Bandinschrift wird ein Binärzähler realisiert, der die Anzahl der 1 und 0 zählt (s_{a1}, s_{a2}, s_{a3}).
- Die abgearbeiteten 1 und 0 werden in e und n umgewandelt (s_1).
- Der Zähler wird durch Löschen der hinteren 0 durch 2 geteilt (s_d) und die Hälfte der e und n werden wieder in 1 und 0 umwandelt (s_{m1}, s_{m2}, s_{m3}).
- Abschließend wird der Zähler gelöscht und die vordere Hälfte der Eingabe wird kopiert ($s_2, s_3, s_4, s_5, s_6, s_7, s_8$).

$$A = (E, B, S, \delta, s_0, F) \text{ mit } E = \{0,1\}, B = \{0,1,n,e,\$,\star\},$$

$$S = \{s_0, s_1, s_{a1}, s_{a2}, s_{a3}, s_d, s_{m1}, s_{m2}, s_{m3}, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_e\}, F = \{s_e\}$$

δ	0	1	$\star$	n	e	\$
s_0	$(s_0, 0, L)$	$(s_0, 1, L)$	$(s_1, \star, R)$	(s_0, n, L)	(s_0, e, L)	
s_1	(s_{a1}, n, L)	(s_{a1}, e, L)	$(s_d, \star, L)$	(s_1, n, R)	(s_1, e, R)	
s_{a1}			$(s_{a2}, \$, L)$	(s_{a1}, n, L)	(s_{a1}, e, L)	$(s_{a2}, \$, L)$
s_{a2}	$(s_{a3}, 1, R)$	$(s_{a2}, 0, L)$	$(s_{a3}, 1, R)$			
s_{a3}	$(s_{a3}, 0, R)$	$(s_{a3}, 1, R)$				$(s_1, \$, R)$
s_d	$(s_{m1}, \$, L)$		$(s_e, \star, N)$	(s_d, n, L)	(s_d, e, L)	$(s_d, \$, L)$
s_{m1}	$(s_{m1}, 1, L)$	$(s_{m2}, 0, R)$	$(s_2, \star, R)$			$(s_{m1}, \$, L)$
s_{m2}	$(s_{m2}, 0, R)$	$(s_{m2}, 1, R)$		$(s_{m3}, 0, L)$	$(s_{m3}, 1, L)$	$(s_{m2}, \$, R)$
s_{m3}	$(s_{m3}, 0, L)$	$(s_{m3}, 1, L)$				$(s_{m1}, \$, L)$
s_2		$(s_2, 1, R)$				$(s_3, \$, R)$
s_3	(s_4, n, R)	(s_5, e, R)	$(s_7, \star, L)$	(s_3, n, R)	(s_3, e, R)	$(s_3, \$, R)$
s_4	$(s_4, 0, R)$	$(s_4, 1, R)$	(s_6, n, L)	(s_4, n, R)	(s_4, e, R)	
s_5	$(s_5, 0, R)$	$(s_5, 1, R)$	(s_6, e, L)	(s_5, n, R)	(s_5, e, R)	
s_6	$(s_6, 0, L)$	$(s_6, 1, L)$		(s_6, n, L)	(s_6, e, L)	$(s_3, \$, R)$
s_7				$(s_7, 0, L)$	$(s_7, 1, L)$	$(s_8, \star, L)$
s_8		$(s_8, \star, L)$	$(s_e, \star, R)$			$(s_8, \star, L)$
s_e			$(s_e, \star, R)$			

b)

$$f'(w) \to \begin{cases} wv, & \text{falls } w = vv \text{ für ein } v \in \{0,1\}^\star \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Die Turingmaschine überprüft also, ob $w = vv$ für ein $v \in \{0,1\}^\star$ ist. In diesem Fall schreibt sie das Wort v noch einmal hinter die Eingabe. Falls die Eingabe nicht die Form vv hat, ist das Verhalten von A' undefiniert. In diesem Fall soll A' wieder in einem Nicht-Endzustand halten. Geben Sie A' vollständig an.

Lösung:

Vorgehensweise:

- Von vorne und hinten abwechselnd die Bandinschrift mit n für 0, e für 1 überschreiben, um die Mitte zu finden (s_0, s_1, s_2, s_3), z. B. $001001 \Rightarrow n0100e \Rightarrow nnenne$.
- Anschließend eine Hälfte mit 0 und 1 überschreiben (s_4), z. B. $nne001$.
- Überprüfen, ob die zwei Hälften identisch sind (s_5, s_6, s_7, s_8), z. B. $nne001 \Rightarrow 001nne$.
- Kopieren einer Hälfte und überschreiben der n mit 0 und der e mit 1 $(s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15})$, z. B. $001nne001 \Rightarrow 001001001$.

$$A = (E, B, S, \delta, s_0, F) \text{ mit } E = \{0,1\}, B = \{0,1,n,e,\star\},$$

$$S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}, s_e\}, F = \{s_e\}$$

δ	0	1	n	e	$\star$
s_0	(s_1, n, R)	(s_1, e, R)	(s_4, n, N)	(s_4, e, N)	
s_1	$(s_1, 0, R)$	$(s_1, 1, R)$	(s_2, n, L)	(s_2, e, L)	$(s_2, \star, L)$
s_2	(s_3, n, L)	(s_3, e, L)			
s_3	$(s_3, 0, L)$	$(s_3, 1, L)$	(s_0, n, R)	(s_0, e, R)	
s_4			$(s_4, 0, R)$	$(s_4, 1, R)$	$(s_5, \star, L)$
s_5	(s_6, n, L)	(s_7, e, L)			$(s_{13}, \star, R)$
s_6	$(s_6, 0, L)$	$(s_6, 1, L)$	$(s_8, 0, R)$		$(s_{14}, \star, R)$
s_7	$(s_7, 0, L)$	$(s_7, 1, L)$		$(s_8, 1, R)$	$(s_{13}, \star, R)$
s_8	$(s_8, 0, R)$	$(s_8, 1, R)$	(s_5, n, L)	(s_5, e, L)	$(s_9, \star, L)$
s_9	$(s_e, 0, N)$	$(s_e, 1, N)$	$(s_{10}, 0, R)$	$(s_{11}, 1, R)$	$(s_e, \star, L)$
s_{10}	$(s_{10}, 0, R)$	$(s_{10}, 1, R)$	(s_{10}, n, R)	(s_{10}, e, R)	$(s_{15}, 0, L)$
s_{11}	$(s_{11}, 0, R)$	$(s_{11}, 1, R)$	(s_{11}, n, R)	(s_{11}, e, R)	$(s_{15}, 1, L)$
s_{12}	$(s_9, 0, R)$	$(s_9, 1, R)$	(s_{12}, n, L)	(s_{12}, e, L)	
s_{13}	$(s_{13}, 0, R)$	$(s_{13}, 1, R)$	$(s_9, 0, R)$	$(s_9, 1, R)$	
s_{14}	$(s_{14}, 0, R)$	$(s_{14}, 1, R)$	$(s_9, 0, R)$	$(s_9, 1, R)$	
s_{15}	$(s_{15}, 0, L)$	$(s_{15}, 1, L)$	(s_{12}, n, N)	(s_{12}, e, N)	$(s_e, \star, R)$
s_e					

Aufgabe 80 ⋆

SPR-AD **Automaten und Sprachen**

Sind folgende Aussagen wahr oder falsch?

Lösung:

	Wahr	Falsch
Alle endlichen Mengen von Wörtern sind Sprachen eines Kellerautomaten.	☒	☐
Jede beliebige Menge von Wörtern ist Sprache einer Turingmaschine.	☐	☒
Zu jedem nichtdeterministischen endlichen Automaten gibt es einen eindeutig bestimmten minimalen deterministischen endlichen Automaten.	☒	☐

Erklärung: Der minimale Automat ist bis auf Isomorphien eindeutig.

Nichtdeterministische endliche Automaten sind mächtiger als deterministische endliche Automaten (in Bezug auf die Sprachmächtigkeit).	☐	☒
Zu jedem nichtdeterministischen endlichen Automaten A gibt es einen regulären Ausdruck α mit $L(\alpha) = L(A)$.	☒	☐

9 Kontextsensitive und monotone Grammatiken

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=354

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 81 ⋆⋆

SPR-AC **Sprachen**

Beantworten Sie, ob die folgenden Aussagen wahr oder falsch sind ($L_i, i \in \{0, 1, 2, 3\}$ steht für die Sprachklasse i in der Chomsky-Hierarchie).

Lösung:

	Wahr	Falsch
$L_1 \subset L_0$	☒	☐
$L_3 = L_{ndet-EA}$	☒	☐
$L_{KA} = L_{ndet-KA}$	☐	☒
$L_{reg} \neq L_3$	☐	☒
$L_0 \neq L_{TM}$	☐	☒
$L_{EA} \neq L_{ndet-EA}$	☐	☒
$L_{ndet-TM} = L_{TM}$	☒	☐

Aufgabe 82 ⋆

SPR-AA **Chomsky-Hierarchie**

Nennen Sie die vier Haupt-Sprachklassen der Chomsky-Hierarchie und die zugehörigen Automatenmodelle und Grammatiken sowie sonstige Charakterisierungen. Geben Sie jeweils eine Beispiel-Grammatik an.

Lösung:

Im Folgenden sei eine Grammatik wie üblich definiert als $G = (N, T, P, S)$.

1) Typ-3-Sprachen (L_3; reguläre Sprachen):
   - Rechtslineare Grammatiken können genau die Sprachen aus L_3 erzeugen. Sie enthalten nur Produktionen der Form $N \times (T \cup TN \cup \{\lambda\})$ in P.
   - Deterministische und nichtdeterministische endliche Automaten können genau die Sprachen aus L_3 erkennen.
   - Durch reguläre Ausdrücke können genau die Sprachen aus L_3 dargestellt werden.
   - Das Pumping-Lemma für reguläre Sprachen ist ein notwendiges Kriterium für jede Sprache aus L_3.
   - Beispiel-Grammatik:

$$\begin{aligned} G &= (\{S, B, C\}, \{a, b, c, d\}, P, S), \\ P &= \{S \to aB, \\ &\quad B \to b \mid dC, \\ &\quad C \to c \mid \lambda\} \end{aligned}$$

2) Typ-2-Sprachen (L_2; kontextfreie Sprachen):
   - Kontextfreie Grammatiken können genau die Sprachen aus L_2 erzeugen. Sie enthalten nur Produktionen der Form $N \times (N \cup T)^\star$ in P.
   - Nichtdeterministische Kellerautomaten können genau die Sprachen aus L_2 erkennen. (Deterministische Kellerautomaten erkennen eine echte Teilmenge der Sprachen aus L_2, die mit den von LR(k)-Grammatiken erzeugbaren Sprachen identisch ist.)
   - Das Pumping-Lemma für kontextfreie Sprachen ist ein notwendiges Kriterium für jede Sprache aus L_2.
   - Beispiel-Grammatik:

$$\begin{aligned} G &= (\{S, A\}, \{a, b\}, P, S), \\ P &= \{S \to aAa \mid aSa, \\ &\quad A \to bAb \mid \lambda\} \end{aligned}$$

3) Typ-1-Sprachen (L_1; kontextsensitive Sprachen):
   - Kontextsensitive Grammatiken können genau die Sprachen aus L_1 erzeugen. Sie enthalten in P nur Produktionen der Form $\varphi_1 A \varphi_2 \to \varphi_1 \psi \varphi_2$, wobei

$$A \in N; \varphi_1, \varphi_2, \psi \in (N \cup T)^\star, \psi \neq \lambda$$

- Darüber hinaus können monotone Grammatiken genau die Sprachen aus L_1 erzeugen. Diese enthalten in P nur Produktionen der Form $\varphi \to \psi$ mit $|\varphi| \leq |\psi|$. (In diesen beiden Grammatiktypen ist ausnahmsweise die Produktion $S \to \lambda$ erlaubt, wenn S sonst auf keiner rechten Seite vorkommt.)
- Nichtdeterministische linear beschränkte Turingmaschinen (LBA) können genau die Sprachen aus L_1 erkennen. Die Frage, ob deterministische linear beschränkte Turingmaschinen ebenfalls dieselbe Klasse von Sprachen erkennen können, ist das bis heute ungelöste "erste LBA-Problem".
- Beispiel-Grammatik:

$$\begin{aligned} G &= (\{S, A, B\}, \{a, b\}, P, S), \\ P &= \{S \to ABS, \\ &\quad abAB \to abaB \mid abba, \\ &\quad AB \to ab, \\ &\quad B \to bb\} \end{aligned}$$

4) Typ-0-Sprachen (L_0; rekursiv aufzählbare/semientscheidbare Sprachen):

- Allgemeine Chomsky-Grammatiken (oder Phrasenstrukturgrammatiken) können genau die Sprachen aus L_0 erzeugen. Ihre Produktionen sind nicht weiter eingeschränkt, außer dass auf der linken Seite nicht ausschließlich Terminalsymbole vorkommen dürfen.
- Deterministische und nichtdeterministische Turingmaschinen können genau die Sprachen aus L_0 erkennen. Allerdings muss eine Turingmaschine, die $L \in L_0$ erkennt, für Wörter $w \notin L$ nicht anhalten. Die von Turingmaschinen **entscheidbaren** Sprachen (für die auch bei Wörtern außerhalb der Sprache ein definiertes Nichtakzeptieren gefordert ist) bilden eine echte Teilmenge von L_0.
- Beispiel-Grammatik:

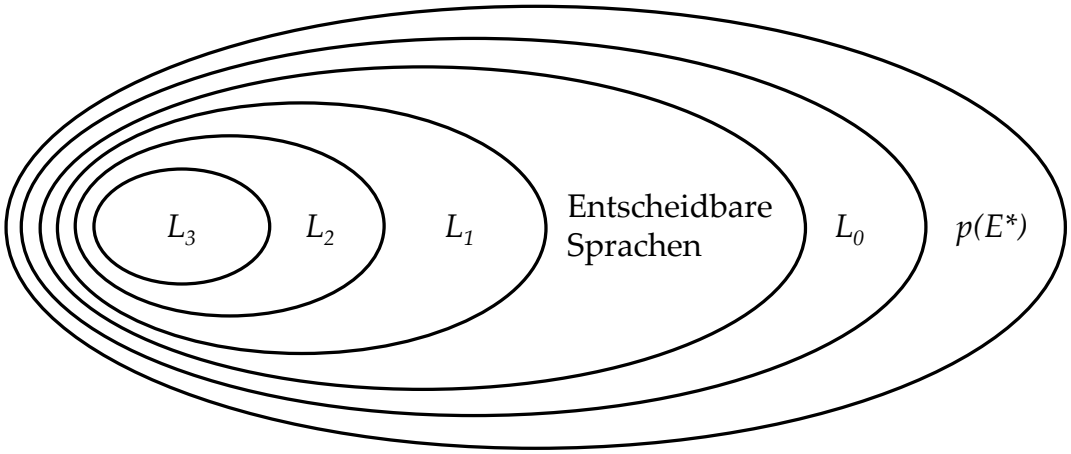
$$\begin{aligned} G &= (\{S, B, C, D\}, \{1, 2\}, P, S), \\ P &= \{S \to BBS \mid BSS \mid \lambda, \\ &\quad BBS \to CD \mid \lambda, \\ &\quad CD \to SS1 \mid 1, \\ &\quad BBBB \to DC \mid 2\} \end{aligned}$$

Aufgabe 83 ★

SPR-AB **Chomsky-Hierarchie**

Zeichnen Sie ein Diagramm, aus dem die Inklusionsbeziehungen der Chomsky-Sprachklassen $L_i, i \in \{0, \ldots, 3\}$, der Menge der entscheidbaren Sprachen und der Menge aller Sprachen $\wp(E^\star)$ über einem Alphabet E deutlich werden.

Lösung:



Aufgabe 84 ★⋆

MON-AB **Monotone/kontextsensitive Grammatiken**

Gegeben sei folgende monotone Grammatik G:

$$
\begin{aligned}
G &= (\{S, T, X, Y\}, \{b\}, P, S), \\
P &= \{S \to YT \mid b \mid bb, \\
&\quad Y \to XY \mid bb, \\
&\quad Xb \to bbX, \\
&\quad XbT \to bbT, \\
&\quad XbbT \to bbbb\}
\end{aligned}
$$

a) Leiten Sie das Wort $w = bbbbbbbb$ mit der Grammatik ab.

Lösung:

Ableitung:

$$S \Rightarrow YT \Rightarrow XYT \Rightarrow XXYT \Rightarrow XXbbT \Rightarrow XbbXbT \Rightarrow bbXbXbT$$
$$\Rightarrow bbbbXXbT \Rightarrow bbbbXbbT \Rightarrow bbbbbbbb$$

b) Welche Sprache wird durch die Grammatik erzeugt?

Lösung:

Offenbar werden die Wörter $b, bb, bbbb, bbbbbbbb$ durch die Grammatik erzeugt ($bbbb$ durch $S \Rightarrow YT \Rightarrow XYT \Rightarrow XbbT \Rightarrow bbbb$). Die Vermutung liegt also nahe, dass

$$L' =_{def} \left\{ b^{2^n} \mid n \in \mathbb{N}_0 \right\} = L(G)$$

Wir stellen fest, dass die ersten beiden Regeln entweder b/bb erzeugen oder Wörter der Form $X^n bbT, n \in \mathbb{N}_0$. Nun müssen alle X und T verschwinden, um terminale Wörter $w \in L(G)$ zu erzeugen. Das geht nur, indem die X von links nach rechts durch $Xb \to bbX$ durchgereicht werden, bis sie über die Regel $XbT \to bbT$ entfernt werden können. Das kann man mit allen X außer dem letzten machen; dieses muss genutzt werden, um über $XbbT \to bbbb$ auch noch das T verschwinden zu lassen. Die Grammatik ist so konstruiert, dass die b's (mit Ausnahme der rechtesten) beim Durchreichen jedes X gerade verdoppelt werden. Mit den Regeln zum Entfernen von X und T ergibt sich letztendlich immer genau eine Zweierpotenz an b's. Wir belassen es bei dieser unformalen Begründung, überlegen Sie bei Interesse selbst, wie man einen echten Beweis führen könnte, dass $L(G) = L'$. Überlegen Sie auch, wie eine nicht-monotone Grammatik für dieselbe Sprache aussehen könnte (diese ist etwas einfacher aufzustellen).

Aufgabe 85 ★

MON-AC **Monotone/kontextsensitive Grammatiken**

Gegeben sei die Sprache

$$L = \{w \in \{a, b, c\}^+ \mid |w|_a = |w|_b = |w|_c\}$$

a) Geben Sie eine monotone Grammatik $G = (N, T, P, S)$ an, so dass gilt $L(G) = L$.

Lösung:

Die monotone Grammatik G lautet:

$$\begin{aligned} G &= (\{S, A, B, C\}, \{a, b, c\}, P, S), \\ P &= \{S \rightarrow ABC \mid ABCS, \\ &\quad AB \rightarrow BA, \\ &\quad AC \rightarrow CA, \\ &\quad BA \rightarrow AB, \\ &\quad BC \rightarrow CB, \\ &\quad CA \rightarrow AC, \\ &\quad CB \rightarrow BC, \\ &\quad A \rightarrow a, \\ &\quad B \rightarrow b, \\ &\quad C \rightarrow c\} \end{aligned}$$

b) Produzieren Sie das Testwort w = bcaabc.

Lösung:

Produktion des Testwortes:

$$S \Rightarrow ABCS \Rightarrow ABCABC \Rightarrow BACABC \Rightarrow BCAABC \Rightarrow^{\star} bcaabc$$

Aufgabe 86 ★⋆

MON-AD **Monotone/kontextsensitive Grammatiken**

Gegeben sei die Sprache

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

a) Geben Sie eine monotone Grammatik G an, so dass gilt $L(G) = L$.

Lösung:

Die monotone Grammatik G lautet:

$$\begin{aligned} G &= (\{S, B, C\}, \{a, b, c\}, P, S), \\ P &= \{S \rightarrow aSBC \mid abC, \\ &\quad CB \rightarrow BC, \\ &\quad bB \rightarrow bb, \\ &\quad bC \rightarrow bc, \\ &\quad cC \rightarrow cc\} \end{aligned}$$

b) Wie groß ist im schlechtesten Fall der Aufwand, um zu überprüfen, ob ein Wort von einer monotonen Grammatik erzeugt werden kann?

Lösung:

Da das Wortproblem für Typ-1-Sprachen $PSPACE$-vollständig ist, benötigt es nach heutigen Kenntnissen im schlechtesten Fall exponentielle Rechenzeit.

Aufgabe 87 $\star^{\star^{\star}}$

MON-AE **Allgemeine und monotone Grammatiken**

In dieser Aufgabe soll die Eigenschaft der Teilbarkeit von Zahlen durch Grammatiken dargestellt werden. Gegeben seien dafür die Sprachen L und $\overline{L}$ mit

$$L = \{a^p \mid p \text{ ist Primzahl}\} \text{ und}$$
$$\overline{L} = \{a\}^\star \backslash L = \{\lambda, a\} \cup \{a^{m \cdot n} \mid m, n \in \mathbb{N}; m, n \geq 2\}$$

Es gilt also:

$$aa, aaa, aaaaa, aaaaaaa, aaaaaaaaaaa, \ldots \in L \ (\notin \overline{L}) \text{ und}$$
$$\lambda, a, aaaa, aaaaaa, aaaaaaaa, \ldots \in \overline{L} \ (\notin L)$$

a) Geben Sie zunächst eine Grammatik $\overline{G}$ an mit $L(\overline{G}) = \overline{L}$.

Lösung:

Die Grammatik $\overline{G}$ lautet:

$$\overline{G} = (\{S, S', A, B, C, X, Y, Z\}, \{a\}, \overline{P}, S)$$
$$\begin{aligned}
\overline{P} = \{ & S \to a \mid S'AAX, \\
& S' \to S'A \mid ZB, \\
& BA \to YAB, \\
& AY \to YA, \\
& BX \to CX \mid \lambda, \\
& AC \to CA, \\
& ZY \to YZ, \\
& ZCA \to ZBA, \\
& A \to a, \\
& Y \to a, \\
& Z \to \lambda\}
\end{aligned}$$

Zunächst wird mit $S' \to S'A$ eine beliebig lange Kette von A's erzeugt. Diese Kette wird dann beliebig oft kopiert.

b) Geben Sie eine Grammatik G an mit $L(G) = L$. Gehen Sie folgendermaßen vor:

- Geben Sie zunächst eine Grammatik G_u für die Sprache L_u an, deren Wörter aus zwei Teilen bestehen, wobei die Anzahl der Zeichen des vorderen Teils die des hinteren **nicht** (restlos) teilt:

$$L_u = \{a^m U_u a^n \mid m, n \in \mathbb{N} : \forall k \in \mathbb{N} : k \cdot m \neq n\}$$

  Hinweis: Es gilt $w \in L_u \Leftrightarrow w = a^n U_u a^m$ mit ($n > m$ oder $a^n U_u a^{m-n} \in L_u$).

- Geben Sie einen **Grammatikteil** an, der aus Strings der Form $X_e a^n Z_e a^m Y_e$ genau dann, wenn $n = m$ gilt, den folgenden String erzeugt: $a^n E_e a^n$ bzw. $a^m E_e a^m$ (beides ist ja derselbe String, wenn $n = m$).
- Erzeugen Sie unter Zuhilfenahme der Teilgrammatiken die Grammatik G.

  Lösung:

  Die Grammatik G_u lautet:

$$\begin{aligned}
G_u &= (N_u, \{a\}, P_u, S_u) \\
N_u &= \{S_u, S'_u, S''_u, A_u, B_u, C_u, U_u, X_u, Y_u, Z_u, E_u\}, \\
P_u &= \{S_u \to X_u C_u C_u S'_u B_u Y_u, \\
&\qquad S'_u \to C_u S'_u B_u \mid C_u S'_u \mid S''_u, \\
&\qquad X_u C_u \to X_u Z_u C_u \mid E_u C_u, \\
&\qquad Z_u C_u \to A_u Z_u, \\
&\qquad Z_u S''_u \to S''_u, \\
&\qquad A_u S''_u B_u \to C_u S''_u B_u B_u, \\
&\qquad A_u C_u \to C_u A_u, \\
&\qquad E_u C_u \to a E_u, \\
&\qquad E_u S''_u \to U_u E_u, \\
&\qquad E_u B_u \to a E_u, \\
&\qquad E_u Y_u \to \lambda\}
\end{aligned}$$

  Die Grammatik funktioniert folgendermaßen:

  - Irgendeine Variante des S-Symbols kodiert das spätere U_u.
  - X_u und Y_u markieren Wortanfang bzw. Wortende.
  - A_u bzw. C_u markieren verschiedene Zustände der Ableitung der vorderen a's, B_u markiert die hinteren a's.
  - Ganz zum Schluss werden diese Platzhalter durch ihre terminale Bedeutung ersetzt; dafür läuft E_u einmal von links nach rechts durch das Wort.

- Mit den zwei oberen Regeln werden alle Strings $\{a^n U_u a^m \mid 1 \leq m < n \in \mathbb{N}\} \subset L'$ (bzw. deren Platzhalter aus Nonterminalen) erzeugt.
- Mit den weiteren Regeln implementiert man die Aussage: Falls $a^n U_u a^m$ ein gültiger String ist, dann auch $a^n U_u a^{m+n}$.

Überlegen Sie selbst, warum auf diese Weise alle Strings aus L_u (und keine anderen) erzeugt werden können.

Anmerkung: U_u sollte eigentlich ein Terminal sein. Da wir die Grammatik aber als Teil der Primzahlgrammatik verwenden wollen, lassen wir es schon hier in der Menge der Nonterminale, wie später benötigt. Dieser Grammatikteil für sich erzeugt somit eigentlich gar keine (terminalen) Wörter.

Der Grammatikteil für Strings der Form $X_e a^n Z_e a^m Y_e$:

$$
\begin{aligned}
N_e = \{&C_e, D_e, E_e, X_e, Y_e, Z_e\},\\
P_e = \{&aZ_e a \to C_e Z_e D_e,\\
&aC_e \to C_e a,\\
&D_e a \to a D_e,\\
&X_e C_e \to a X_e,\\
&D_e Y_e \to Y_e a,\\
&X_e Z_e Y_e \to E_e\}
\end{aligned}
$$

Die Grammatik G:

$$
\begin{aligned}
G = (&N_u \cup N_e \cup \{S, C, D, X, Y, X_0, Y_0, K, K', T, V\}, \{a\}, P, S) \\
P = &P_u \cup P_e \cup \\
&\{S \to X_0 XaaTaaaY \mid aa, \\
&T \to Taa \mid aV \mid K, \\
&V \to X_t S_u Y_t, \\
&Xa \to XX_e a, \\
&aY \to aY_e Y, \\
&U_u \to Y_e U'_u X_e, \\
&X_t \to Z_e, \\
&Y_t \to Z_e, \\
&aU'_u \to U'_u, \\
&U'_u a \to U'_u, \\
&E_e U'_u E_e \to aV \mid K, \\
&aKa \to CKD, \\
&aC \to Ca, \\
&Da \to aD, \\
&XC \to aX, \\
&DY \to Ya, \\
&XKaY \to K'a, \\
&aK'a \to K'a, \\
&X_0 K'a \to a\}
\end{aligned}
$$

Die Grammatik erzeugt "aa" oder ein Wort $X_0 XaaTa^n Y, n$ ungerade. Die Bedeutung des letzteren ist in der Folge, dass der vordere Teil durch die Anzahl k der "a"s angibt, bis zu welcher Zahl der hintere Teil a^p **nicht** teilbar ist. Um zu zeigen, dass der hintere Teil prim ist, wird der vordere Teil sukzessive um "a" erweitert; bei jeder Erweiterung wird über S_u irgendein Wort erzeugt, bei dem der vordere Teil den hinteren "nicht" teilt (man kann sich ja nicht aussuchen, welches Wort erzeugt wird, sondern muss alle Möglichkeiten beachten). Dann wird mit der Gleichheits-Grammatik überprüft, ob das erzeugte Wort genau der Aussage entspricht, dass der neue vordere Teil $k+1$ den hinteren Teil p nicht teilt. In diesem Fall weiß man, dass der hintere Teil bis zu $k+1$ nicht teilbar ist. Wenn man bei der Aussage "p ist bis $p-1$ nicht teilbar" angekommen ist, kann man mit K den Abschlussvorgang starten. Es ist wichtig zu beachten, dass in allen anderen Fällen keine komplett terminalen Wörter abgeleitet werden können (bspw. wenn K zu früh erzeugt wird oder wenn aus dem Nonterminal S_u ein unpassendes Wort erzeugt wird; und insbesondere wenn p gar nicht prim ist).

c) Leiten Sie $aaaaaaaaa \in \overline{L}$ aus $\overline{G}$ ab. Schätzen Sie, wie viele Ableitungsschritte man

bräuchte, um $aaaaa \in L$ aus G abzuleiten.

Lösung:

- $aaaaaaaaa$ aus $\overline{G}$:

$$\begin{aligned}
&\overline{S} \Rightarrow \overline{S'}AAX \Rightarrow \overline{S'}AAAX \Rightarrow Z\overline{BA}AAX \Rightarrow ZYA\overline{BA}AX \Rightarrow ZYAYA\overline{BA}X \\
&\Rightarrow ZY\overline{AY}AYABX \Rightarrow ZYYA\overline{AY}ABX \Rightarrow ZYY\overline{AY}AABX \Rightarrow ZYYYAAA\overline{BX} \\
&\Rightarrow ZYYYAA\overline{AC}X \Rightarrow ZYYYA\overline{AC}AX \Rightarrow ZYYY\overline{AC}AAX \Rightarrow \overline{ZY}YYCAAAX \\
&\Rightarrow Y\overline{ZY}YCAAAX \Rightarrow YY\overline{ZY}CAAAX \Rightarrow YYY\overline{ZC}AAAX \Rightarrow YYYZ\overline{BA}AAX \\
&\Rightarrow YYYZYA\overline{BA}AX \Rightarrow YYYZYAYA\overline{BA}X \Rightarrow YYYZY\overline{AY}AYABX \\
&\Rightarrow \overline{YYYZYAYAYA} \Rightarrow^{\star} aaaaaaaaa.
\end{aligned}$$

- $aaaaa$ aus G: Die Ableitung benötigt mit der hier vorgeschlagenen Grammatik G etwa 300 Ableitungsschritte.

d) Sind Ihre Grammatiken G und $\overline{G}$ kontextsensitiv bzw. monoton? Falls nicht, argumentieren Sie, warum es kontextsensitive bzw. monotone Versionen G_1 bzw. $\overline{G}_1$ zu beiden geben muss.

Lösung:

Die beiden angegebenen Grammatiken sind nicht monoton ($\overline{G}$ wegen der beiden λ-Regeln, G wegen mehrerer verkürzender Regeln.) Es ist jedoch leicht (zumindest prinzipiell), sich eine linear beschränkte Turingmaschine zu überlegen, die L bzw. $\overline{L}$ erkennt. Damit muss es auch Typ-1-Grammatiken für diese Sprachen geben. Mit einem entsprechenden Algorithmus kann man die Grammatiken direkt aus den Turingmaschinen generieren.

Aufgabe 88 ★⋆

SPR-AF **Allgemeine Grammatiken**

Schreiben Sie ein Programm in Pseudocode, das bei Eingabe einer beliebigen Grammatik alle Wörter aufzählt, die von der Grammatik erzeugt werden. Die Grammatik ist gegeben durch

- eine Menge von Nonterminalen $N = \{n_1, \ldots, n_k\}$,
- eine Menge von Terminalen $T = \{t_1, \ldots, t_l\}$,
- eine Menge von Regeln $P = \{(l_1 \to r_1), \ldots, (l_n \to r_n)\}$ mit $l_i, r_i \in (N \cup T)^{\star}$ $(l_i \notin T^{\star})$,

- ein Startsymbol $S \in N$.

Sie dürfen im Pseudocode folgende Elemente verwenden:

- Die üblichen Programmierkonstrukte wie `if B then ... (else ...) end if`, `while B loop ... end loop`, `for all x in X loop ... end loop`, ...
- Die üblichen mathematischen Operationen wie $x \in X$, $X \cup Y$, ...
- Eine Methode mit Signatur `match(w, p)`, die überprüft, ob die Regel $p \in P$ auf das Wort $w \in (N \cup T)^\star$ angewendet werden kann.
  - Es wird die (potenziell leere) Menge $\{w_1, \ldots, w_k\}$ aller Wörter zurückgegeben, die beim Anwenden von p auf w entstehen können.
  - Beispiel: $match(\underbrace{(a, S, b)}_{w}, \underbrace{((S, b) \to (a, c, b))}_{p}) = \{\underbrace{(a, a, c, b)}_{w_1}\}$
- Eine Methode `isTerminal(w)`, die wahr zurückgibt, gdw. $w \in T^\star$.

Lösung:

```
Set W = {S};

while W != {} loop
  Set A = {};

  for all w in W loop
    if isTerminal(w) then
      print(w);
    else
      for all p in P loop
        A = A + match(w, p);
      end loop
    end if
  end loop

  W = A;
end loop

print("Fertig. Die Sprache der Grammatik ist endlich.");
```

10 Berechenbarkeits- und Komplexitätstheorie

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=356

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

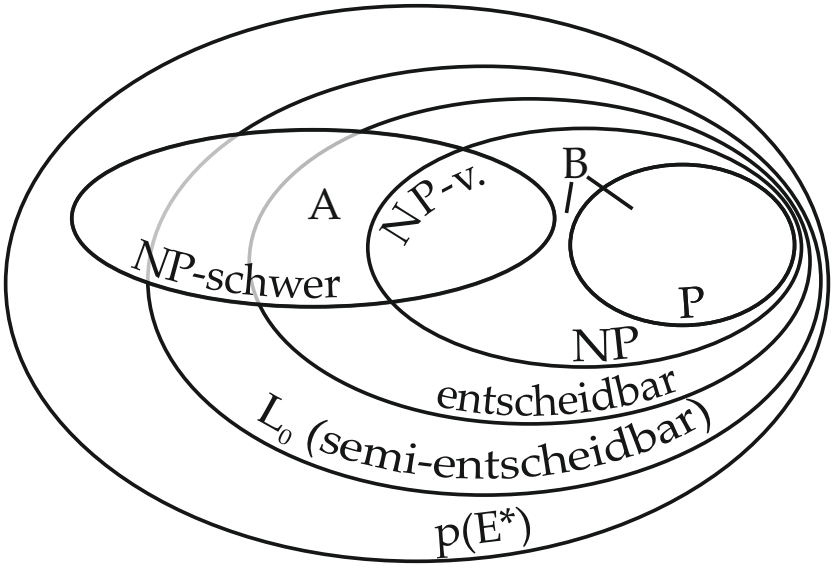
«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 89 ★⋆

BER-AA **Komplexitäts- und Sprachklassen**

Betrachten Sie die folgende Abbildung:

Lösung:



a) Tragen Sie (unter der Annahme $P \neq NP$ und Berücksichtigung der Inklusionsbeziehungen) folgende Problemklassen in die Abbildung ein:
   - P,
   - NP,
   - NP-vollständig,
   - NP-schwer,
   - Entscheidbare Probleme,
   - Semientscheidbare Probleme,
   - Menge aller Entscheidungsprobleme $\wp(E^\star)$.

b) Wo würde ein Problem A liegen, welches zwar NP-schwer und entscheidbar, nicht jedoch NP-vollständig ist? Wo könnte ein Problem $B \in NP$ liegen, welches nicht NP-schwer ist?

c) Welche Eigenschaften hat ein Problem X, das zur Klasse der NP-schweren Probleme gehört?

Lösung:

$\forall Y \in NP : Y \leq_{pol} X$ (oder $\exists Y \in NP$-vollständig: $Y \leq_{pol} X$)

d) Welche Eigenschaften hat ein Problem X, das zur Klasse der NP-vollständigen Probleme gehört?

Lösung:

X ist NP-schwer und $X \in NP$

e) Was würde sich in der Abbildung verändern, wenn man von der Annahme $P = NP$ ausginge?

Lösung:

Die Bereiche für P und NP würden zusammenfallen. Der Bereich NP-vollständig würde sich fast damit überdecken, es gäbe technisch gesehen aber noch die beiden (trivialen) Probleme $\emptyset$ und $E^\star$, die in $P = NP$ wären, aber nicht NP-vollständig.

f) Welche prinzipiellen Vorgehensweisen sind denkbar, wenn man beweisen möchte, dass:

- P = NP,
- P ≠ NP.

Lösung:

- $P = NP$: Man gibt für ein beliebiges NP-vollständiges Problem einen deterministischen Polynomialzeit-Algorithmus an oder man führt einen Beweis, der nicht konstruktiv ist, also keinen Algorithmus liefert.
- $P \neq NP$: Man führt einen nicht-konstruktiven Beweis, indem man zeigt, dass es für ein NP-vollständiges Problem (bzw. für irgendein Problem aus NP) keinen deterministischen Polynomialzeit-Algorithmus gibt.

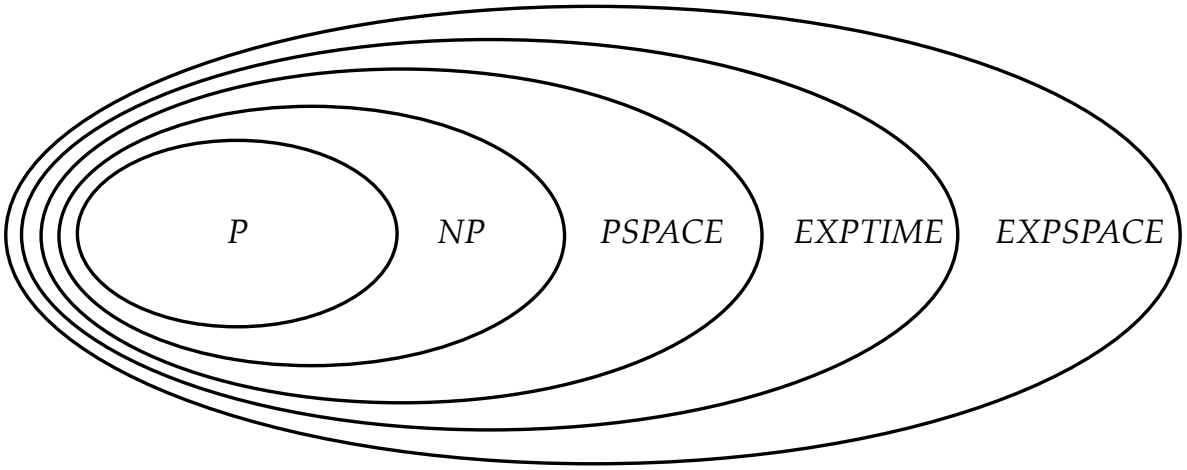
Aufgabe 90 ⋆

BER-AJ **Komplexitätsklassen**

Ordnen Sie die Klassen $PSPACE$, NP, P, $EXPSPACE$ und $EXPTIME$ in einem Schaubild an, welches die untereinander geltenden Inklusionsbeziehungen darstellt. Von welcher dieser Klassen ist bekannt, dass sie echt unterschiedlich sind?

Lösung:

Es gilt folgende Teilmengenbeziehung (wobei meist unbekannt ist, ob die Grenzen echte Inklusionen darstellen oder die Mengen doch identisch sind):



Es ist lediglich bekannt, dass $P \subsetneq EXPTIME$ und $PSPACE \subsetneq EXPSPACE$. Im Allgemeinen wird jedoch angenommen, dass alle diese Mengen unterschiedlich sind.

Aufgabe 91 ⋆⋆

BER-AB **Komplexitätsklassen und Polynomialzeitreduktion**

Es gelte für die Probleme A, B, C, D:

- A ist NP-vollständig,
- $B \in NP$,
- C ist NP-schwer,
- $D \in P$.

Wenn nichts anders vermerkt ist, gelte die Annahme $P \neq NP$. Geben Sie an, welche der folgenden Aussagen aus diesen Annahmen folgen.

Lösung:

	Wahr	Falsch
Eine Lösung für A kann in Polynomialzeit verifiziert werden.	☒	☐
B kann nicht in Polynomialzeit gelöst werden.	☐	☒
Falls $P = NP$, dann ist C in Polynomialzeit lösbar.	☐	☒
Es gilt $A \leq_{pol} C$.	☒	☐
Für ein Problem F gilt: $F \in EXPTIME \Rightarrow F \in EXPSPACE$.	☒	☐
Jede abzählbare Menge ist auch entscheidbar.	☐	☒
D ist in Polynomialzeit reduzierbar auf C, aber nicht auf A.	☐	☒

Erklärung: D ist sowohl auf C als auch auf A in Polynomialzeit reduzierbar.

Aufgabe 92 ★⋆

BER-AH **Polynomialzeitreduktion**

Gegeben seien SAT (das NP-vollständige Erfüllbarkeitsproblem der Aussagenlogik) sowie die Probleme A, B, C, D und E mit den folgenden Eigenschaften:

- $A \in NP$-vollständig,
- $B \in NP$,
- $C \in NP$-schwer,
- $D \in P$,
- E ist entscheidbar.

Entscheiden Sie, welche der folgenden Aussagen aus den Angaben folgen und welche nicht (unter der Annahme $P \neq NP$).

Lösung:

	Wahr	Falsch
$A \leq_{pol} SAT$	☒	☐
$A \leq_{pol} C$	☒	☐
$A \leq_{pol} D$	☐	☒
$B \leq_{pol} C$	☒	☐
$E \leq_{pol} C \Rightarrow E \in NP$	☐	☒
A ist nicht entscheidbar	☐	☒
B ist entscheidbar	☒	☐
C ist entscheidbar	☐	☒
D ist entscheidbar	☒	☐
A ist semientscheidbar	☒	☐
$A \leq_{pol} D \Rightarrow P = NP$	☒	☐
C ist in Polynomialzeit lösbar	☐	☒
$A \in EXPTIME$	☒	☐
$A \notin PSPACE$	☐	☒
$A \notin EXPTIME \setminus PSPACE$	☒	☐

Aufgabe 93 ⋆⋆

BER-AI **Polynomialzeitreduktion**

Gegeben sei das Problem SAT und die drei Probleme A, B und C, für die Folgendes gilt:

- A ist ein entscheidbares Problem,
- B ist NP-schwer und nichtdeterministisch in Polynomialzeit lösbar,
- C ist in der Menge P.

Geben Sie an, ob die folgenden Aussagen aus den Annahmen folgen oder nicht. Nehmen Sie dabei an, dass $P \neq NP$ gilt.

Lösung:

	Wahr	Falsch
Wenn A NP-schwer ist, so kann es in Polynomialzeit auf jedes NP-vollständige Problem reduziert werden.	☐	☒
Wenn $A \in NP$, dann ist A auch NP-vollständig.	☐	☒
B lässt sich auf C reduzieren.	☐	☒
B lässt sich auf SAT reduzieren.	☒	☐
Gilt entgegen der Annahme $P = NP$, dann ist C NP-vollständig.	☐	☒

Erklärung: Das gilt zwar für fast alle Probleme $C \in P$, allerdings nicht für die Spezialfälle $C = E^\star$ bzw. $C = \emptyset$; daher ist die Aussage falsch.

Das Problem "ist $n \in \mathbb{N}$, eine Primzahl?" ist entscheidbar.	☒	☐
Jede entscheidbare Menge ist auch aufzählbar.	☒	☐

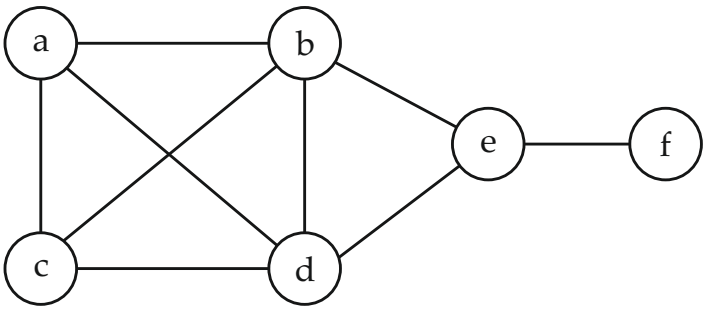
Aufgabe 94 ★★⋆

BER-AC **Polynomialzeitreduktion**

Eine Knotenabdeckung der Größe k eines Graphen $G = (V, E)$ ist eine Teilmenge der Knoten $C \subseteq V$ mit $|C| = k$, sodass jede Kante mit mindestens einem Knoten aus C verbunden ist. Das Problem VC (Vertex Cover) ist die Frage, ob es zu einem Graphen $G = (V, E)$ eine Knotenabdeckung einer bestimmten Größe k gibt. Formal:

$$VC = \{(G, k) \mid \exists C \subseteq V : |C| = k \wedge \forall (v_1, v_2) \in E : v_1 \in C \vee v_2 \in C\}$$

Beispielsweise wäre im abgebildeten Graphen eine (in diesem Fall minimale) Knotenabdeckung $C = \{a, b, c, e\}$ mit $k = 4$.



a) Zeigen Sie: $VC \in NP$.

Lösung:

Für eine geratene Knotenteilmenge $C \subseteq V$ mit $|C| = k$ kann in Polynomialzeit verifiziert werden, ob alle Kanten einen Knoten in C haben.

b) Zeigen Sie, dass VC NP-vollständig ist, indem Sie das Problem $CLIQUE$ darauf reduzieren.

Lösung:

Behauptung: Für $G = (V, E), k \in \mathbb{N}$ gilt

$$(G, k) \in CLIQUE \Leftrightarrow (G^c, |V| - k) \in VC$$

wobei $G^c = (V, E')$ und $E' = \{(v_1, v_2) \in V \times V \mid (v_1, v_2) \notin V\}$. In G^c gibt es zwischen zwei Knoten also genau dann eine Kante, wenn es in G keine gibt.

Beweis:

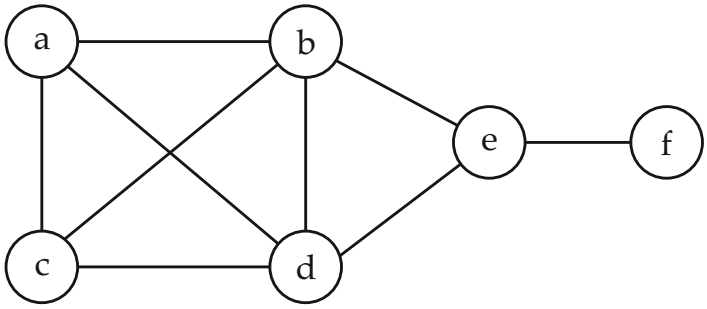
$$
\begin{aligned}
&(G,k) \in CLIQUE \\
\Leftrightarrow &\exists S \subseteq V, |S| = k : \forall v_1, v_2 \in S : (v_1, v_2) \in E \\
&(S \text{ enthält } k \text{ Knoten und alle Knoten aus } S \text{ sind in } G \text{ paarweise verbunden}) \\
\Leftrightarrow &\exists S \subseteq V, |S| = k : \forall v_1, v_2 \in S : (v_1, v_2) \notin E' \\
&(\text{alle Knoten aus } S \text{ sind in } G^c \text{ paarweise unverbunden}) \\
\Leftrightarrow &\exists S \subseteq V, |S| = k : \forall (v_1, v_2) \in E' : v_1 \in V \backslash S \vee v_2 \in V \backslash S \\
&(V \backslash S \text{ ist Knotenüberdeckung von } G^c \text{ mit Größe } |V| - k) \\
\Leftrightarrow &(G^c, |V| - k) \in VC
\end{aligned}
$$

Da die Umformung von (G,k) nach $(G^c, |V|-k)$ in Polynomialzeit möglich ist, gilt damit $CLIQUE \leq_{pol} VC$ und mit dem Ergebnis aus a), dass VC NP-vollständig ist.

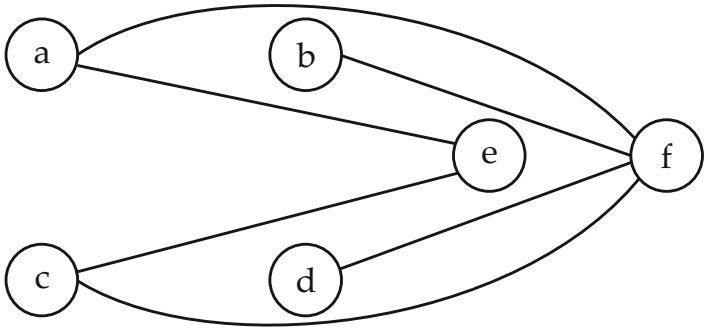
□

Beispiel: Für $k = 4$ bilden in der Abbildung die Knoten a, b, c, d die Menge S, e, f sind in der Menge $V \backslash S$.

G:



G^c:



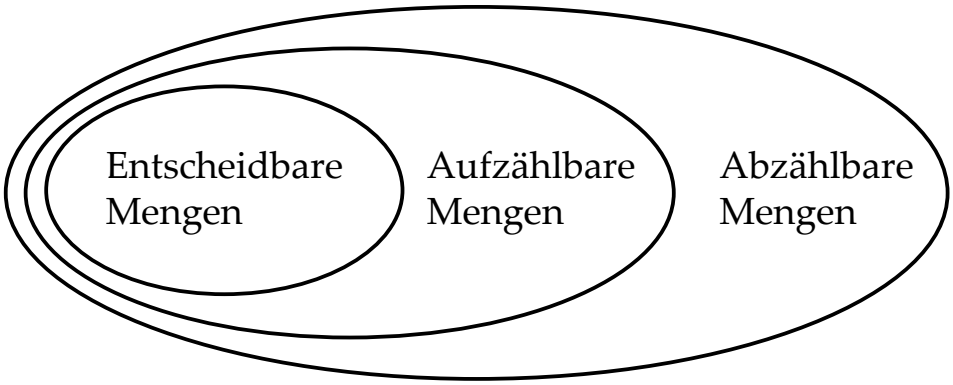
Aufgabe 95 ⋆

BER-AK **Abzählbarkeit, Aufzählbarkeit, Entscheidbarkeit**

a) Beschriften Sie das vorgezeichnete Diagramm mit den Bezeichnungen
   - abzählbare Mengen,
   - entscheidbare Mengen und
   - aufzählbare Mengen,

   sodass die zugehörigen Inklusionsbeziehungen widergespiegelt werden.

   Lösung:



b) Was bedeutet es, wenn eine Turingmaschine eine Funktion "berechnet"?

   Lösung:

   Eine Funktion $f : M \subseteq E^\star \to N \subseteq B^\star$, wobei E und B Alphabete sind, heißt genau dann Turing-berechenbar, wenn es eine Turingmaschine $T = (E, B, S, \delta, s_0, F)$ gibt, die f "berechnet". Das bedeutet, dass die Rechnung der Turingmaschine, wenn zu Beginn ein Wort $w \in M$ (umgeben von $\star$) auf dem Band steht, in endlicher Zeit terminiert und nach Terminierung das Wort $f(w) \in N$ (umgeben von $\star$) auf dem Band steht. Für Wörter außerhalb des Definitionsbereichs von f muss die Turingmaschine weder eine bestimmte Ausgabe liefern noch halten.

c) Nennen Sie die Definition von Turing-Aufzählbarkeit.

   Lösung:

Eine Menge $M \subseteq E^\star$ für ein Alphabet E heißt genau dann Turing-aufzählbar, wenn es eine surjektive, Turing-berechenbare Funktion $f : \mathbb{N} \to M$ gibt. Man kann also in endlicher Zeit herausfinden, dass eine Eingabe $w \in E^\star$ in der Menge M enthalten ist, indem man nacheinander $f(1), f(2), \ldots$ betrachtet oder "aufzählt". Wenn für ein $n \in \mathbb{N}$ gilt: $f(n) = w$, dann weiß man, dass $w \in M$. Falls w jedoch nicht in M ist, hat man keine Möglichkeit, das herauszufinden, da die beschriebene Prozedur dann endlos läuft, man aber nicht weiß, ob für ein höheres n nicht doch noch $f(n) = w$ gelten wird. In der Praxis ist diese Methode also im Allgemeinen nicht verwendbar.

d) Nennen Sie die Definition von Turing-Entscheidbarkeit.

Lösung:

Eine Menge $M \subseteq E^\star$ für ein Alphabet E heißt genau dann Turing-entscheidbar, wenn ihre charakteristische Funktion χ_M Turing-berechenbar ist. Insofern sind die Begriffe Entscheidbarkeit und Berechenbarkeit in gewisser Weise äquivalent.

e) Warum nennt man Turing-aufzählbare Mengen auch semientscheidbare Mengen?

Lösung:

Eine Menge M ist genau dann Turing-aufzählbar, wenn es eine Turingmaschine gibt, die genau die Eingaben $w \in M$ akzeptiert. Für $w \notin M$ jedoch kann es sein, dass die Turingmaschine nie stoppt (siehe Aufgabenteil c) und die entsprechende Erklärung). Da solche Mengen sozusagen nur "zur Hälfte" entschieden werden, nennt man sie semientscheidbar.

f) Geben Sie die Idee einer universellen Turingmaschine wieder und erklären Sie deren grundlegende Funktionsweise.

Lösung:

Eine "gewöhnliche" Turingmaschine T berechnet genau eine Funktion T_f. Eine universelle Turingmaschine U soll andere Turingmaschinen simulieren können, d. h. verschiedene Funktionen berechnen können, je nachdem, was für eine Turingmaschine eingegeben wird. Damit soll durch das Automatenmodell die Verbindung zu realen Computern, die beliebige Programme ausführen können, hergestellt werden, um damit die Turingmaschine als deren theoretische Grundlage zu rechtfertigen. Hierzu erhält U als Teil der Bandeingabe zunächst eine Kodierung $[T]$ der jeweiligen Turingmaschine T, deren Funktion sie berechnen soll; dazu gehört insbesondere eine Kodierung aller Zustandsübergänge. Davon abgetrennt steht in der Eingabe von U eine Kodierung $[(\lambda, s_0, w)]$ der Anfangskonfiguration von T, also insbesondere auch des Eingabewortes, auf dem T simuliert werden soll. Nun wird auf dem Arbeitsband fortlaufend eine Kodierung der Folgekonfiguration der aktuell auf dem Band stehenden Konfiguration

entsprechend der kodierten Arbeitsweise von T erzeugt. Wenn U während dieser Prozedur die Kodierung einer Endkonfiguration von T erzeugt hat, geht auch die universelle Turingmaschine in eine Endkonfiguration über. Die Bandbeschriftung kann zu Anfang etwa wie folgt aussehen (wobei \$ als Trennzeichen dient):

$\cdots$	$\star$	$[T]$	\$	$[(\lambda, s_0, w)]$	$\star$	$\cdots$

Aufgabe 96 $\star^{\star\star}$

BER-AD **Diagonalisierung, Überabzählbarkeit**

Zeigen Sie durch Diagonalisierung folgende Aussagen:

a) Für ein Alphabet E ist $\wp(E^\star)$ (also die Menge aller Sprachen über den Wörtern aus $E^\star$) überabzählbar.

Lösung:

Die Wörter aus $E^\star$ können abgezählt werden, bspw. durch eine längenlexikographische Sortierung: $w_1 = \lambda, w_2 = a, w_3 = b, w_4 = aa, w_5 = ab, w_6 = ba, w_7 = bb, \ldots$ für das Alphabet $\{a, b\}$. Wir können sie also als unendlich lange, aber abzählbare Liste anordnen.

Wir nehmen an, dass die Potenzmenge von $E^\star$ ebenfalls abgezählt werden kann. Dann kann jeder Teilmenge $M \subseteq E^\star$ (also $M \in \wp(E^\star)$) eindeutig eine natürliche 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 $\wp(E^\star)$ in der Liste enthalten ist.

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

	w_1	w_2	w_3	w_4	w_5	$\ldots$
M_1	0	0	1	1	0	$\ldots$
M_2	1	0	1	0	0	$\ldots$
M_3	1	0	1	1	1	$\ldots$
M_4	0	1	1	1	0	$\ldots$
$\vdots$						

In jeder Zelle T_{ij} 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

$$M' = \{w_j \mid w_j \notin M_j\} \subseteq E^\star \text{ mit } j \in \mathbb{N}$$

In obigem Beispiel enthielte 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 $E^\star$ 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 $\wp(E^\star)$ ist nicht abzählbar, sondern überabzählbar.

□

b) Die Menge $\mathbb{R}$ der reellen Zahlen ist überabzählbar.

Lösung:

Wir zeigen, dass sogar das Intervall $[0, 1[$ überabzählbar ist, was die obige Behauptung impliziert. Wir nehmen an, dass die Zahlen aus $[0, 1[$ abgezählt werden können, dass sie also wie in a) durch eine Liste $(r_1, r_2, r_3, \ldots)$ unendlicher, aber abzählbarer Länge vollständig angegeben werden können.

Wir stellen folgende Tabelle T der reellen Zahlen zwischen 0 und 1 auf (beispielhaft ausgefüllt):

	d_1	d_2	d_3	d_4	d_5	$\ldots$
r_1	0	5	7	1	4	$\ldots$
r_2	7	1	4	5	1	$\ldots$
r_3	3	5	7	6	2	$\ldots$
r_4	3	2	6	9	5	$\ldots$
$\vdots$						

Eine Zeile i in der Tabelle definiert nun die Nachkommastellen der reellen Zahl $0 \leq r_i < 1$; eine Zelle T_{ij} gibt demnach die j-te Nachkommastelle der i-ten reellen Zahl r_i an. Bspw. gilt $r_1 = 0{,}05714\ldots$ Betrachte die Zahl r', die sich aus der Hintereinanderschreibung

$$r' = 0, \cdot \prod_{i=1}^{\infty} (T_{ii} + 1) \mod 10$$

ergibt (das Produkt wird hier als Symbol für Konkatenation verwendet wie bei regulären Ausdrücken, der Nachkommateil der Zahl r' ist also gerade die Diagonale der Tabelle um eins inkrementiert und modulo 10 genommen). In obigem Beispiel wäre $r' = 0{,}1280\ldots$

Unserer Annahme folgend müsste $r' = r_n$ für ein $n \in \mathbb{N}$ gelten. Nach Definition unterscheidet sich r' aber an mindestens einer Stelle (der Diagonalstelle) von allen Zahlen in der Liste. Da es (mind.) eine Zahl gibt, die in unserer Liste nicht vorkommt, muss die Annahme falsch sein und es gibt überabzählbar viele Zahlen in $[0, 1[$ und damit auch in $\mathbb{R}$.

□

Aufgabe 97 ⋆⋆

BER-AG **Berechenbarkeit**

a) Wann ist eine Funktion Turing-berechenbar?

Lösung:

Gegeben seien ein Alphabet E und zwei Mengen $M_1, M_2 \subseteq E^\star$ von Wörtern über E. Eine Funktion $f : M_1 \to M_2$ heißt Turing-berechenbar, wenn es eine Turingmaschine $T = (E, B, S, \delta, s_0, F)$ mit $E^\star \supseteq M_1$ und $B^\star \supseteq M_2$ gibt, die eine Funktion $T_f : M_1 \to M_2$ berechnet mit $\forall w \in M_1 : T_f(w) = f(w)$.

b) Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind.

Lösung:

	Wahr	Falsch
Jede Funktion ist berechenbar.	☐	☒
Die Menge aller berechenbarer Funktionen ist abzählbar.	☒	☐
Wenn eine Menge M abzählbar ist, dann ist sie auch aufzählbar.	☐	☒
Die Menge $M = \{n, n+1, n+2, \ldots, n+n\}$ für $0 < n < \infty$ ist überabzählbar.	☐	☒
Das reelle Intervall $[3, 6]$ ist abzählbar.	☐	☒

c) Geben Sie jeweils ein Beispiel an für Mengen, die

- entscheidbar,
- aufzählbar aber nicht entscheidbar,
- abzählbar aber nicht aufzählbar sind.

Lösung:

- **Entscheidbar**: Menge der Primzahlen, Menge der Quadratzahlen, Menge der Palindrome usw.

- **Aufzählbar aber nicht entscheidbar**: Die Menge der Kodierungen aller Turingmaschinen mit einer zugehörigen Eingabe, die auf dieser Eingabe anhalten.
- **Abzählbar aber nicht aufzählbar**: Die Menge der Kodierungen aller Turingmaschinen, die auf ihrer eigenen Kodierung als Eingabe nicht halten.

d) Ist die Menge $\left\{-17, -3, \frac{1}{4}, 2, 73, 84, 15^2\right\}$ abzählbar, aufzählbar bzw. entscheidbar?

Lösung:

Da es sich um eine endliche Menge handelt, ist diese abzählbar, aufzählbar und entscheidbar.

e) Geben Sie einen Pseudocode-Algorithmus an, der die Menge der Quadratzahlen aufzählt.

Lösung:

```
int i := 0;
forever do
    print (i·i)
    i := i + 1;
end do
```

Aufgabe 98 ⋆⋆

BER-AE **Berechenbarkeit**

Sei E ein Alphabet. Beweisen Sie, dass nicht jede Funktion $f : E^\star \to E^\star$ berechenbar ist.

Lösung:

Wir zeigen die Behauptung über die folgenden zwei Lemmata:

1) Die Menge aller Funktionen $f : E^\star \to E^\star$ ist überabzählbar.
2) Die Menge aller berechenbaren Funktionen $f : E^\star \to E^\star$ ist abzählbar.

Daraus folgt, dass nicht jede Funktion berechenbar sein kann, da es mehr Funktionen als berechenbare Funktionen gibt. Beweis:

1) Wir nehmen probehalber an, dass die Menge F aller Funktionen $f : E^\star \to E^\star$ abzählbar ist. Dann kann man die Elemente von F in einer unendlich langen, aber abzählbaren Liste anordnen:

$$F = \{f_1, f_2, \ldots\}$$

Wie auf Seite 135 gezeigt, können die Wörter aus $E^\star$ abgezählt werden, bspw. durch eine längenlexikographische Sortierung. Man kann also $E^\star$ schreiben als:

$$E^\star = \{w_1, w_2, \ldots\}$$

Wenn die Annahme stimmt, lassen sich daher alle Funktionen mitsamt ihrer Funktionswerte in einer unendlich großen, aber abzählbaren Tabelle darstellen:

	w_1	w_2	w_3	w_4	$\ldots$
f_1	$f_1(w_1)$	$f_1(w_2)$	$f_1(w_3)$	$f_1(w_4)$	$\ldots$
f_2	$f_2(w_1)$	$f_2(w_2)$	$f_2(w_3)$	$f_2(w_4)$	$\ldots$
f_3	$f_3(w_1)$	$f_3(w_2)$	$f_3(w_3)$	$f_3(w_4)$	$\ldots$
$\vdots$					

Definiere nun für $u, v \in E^\star, u \neq v$ eine Funktion $g : E^\star \to E^\star$ durch

$$\forall i \in \mathbb{N} : g(w_i) = \begin{cases} u, & \text{falls } f_i(w_i) \neq u \\ v, & \text{falls } f_i(w_i) = u \end{cases}$$

Die Funktion g unterscheidet sich also von allen Funktionen in der Tabelle in mindestens einem Funktionswert. Wenn F abzählbar ist, muss es ein $k \in \mathbb{N}$ geben, sodass $g = f_k$. Dann muss insbesondere auch gelten, dass $g(w_k) = f_k(w_k)$. Nach Definition von g gilt aber $g(w_k) \neq f_k(w_k)$. Wir kommen also zu einem Widerspruch und müssen die ursprüngliche Annahme aufgeben, dass F abzählbar ist.

2) Dass die Menge aller berechenbaren Funktionen F_b abzählbar ist, sieht man durch folgende (informale) Überlegung: Wenn $f \in F_b$ berechenbar ist, dann gibt es einen Text endlicher Länge über einem Alphabet E zur Beschreibung eines f berechnenden Algorithmus. Da die Menge aller Texte $E^\star$ abzählbar ist, ist die Menge aller berechenbaren Funktionen abzählbar.

Wie oben beschrieben, folgern wir aus den Lemmata, dass es mehr Funktionen als berechenbare Funktionen gibt und dass somit nicht alle Funktionen $f : E^\star \to E^\star$ berechenbar sind.

□

Aufgabe 99 ★

BER-AL **Berechenbarkeits- und Komplexitätstheorie**

Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind.

Lösung:

	Wahr	Falsch
Nicht jede Funktion ist berechenbar.	☒	☐
Wenn eine Menge M abzählbar ist, dann ist M auch aufzählbar.	☐	☒
Die Äquivalenz zweier beliebiger Programme A und B ist entscheidbar.	☐	☒
Das Travelling-Salesman-Problem liegt in $PSPACE$.	☒	☐
$Q \leq_{pol} P \wedge Q$ ist in Polynomialzeit lösbar $\Rightarrow$ P ist in Polynomialzeit lösbar.	☐	☒
$Q \leq_{pol} P \wedge P$ ist in Polynomialzeit lösbar $\Rightarrow$ Q ist in Polynomialzeit lösbar.	☒	☐
$Q \leq_{pol} P \wedge Q$ ist nicht in Polynomialzeit lösbar $\Rightarrow$ P ist nicht in Polynomialzeit lösbar.	☒	☐
$Q \leq_{pol} P \wedge P$ ist nicht in Polynomialzeit lösbar $\Rightarrow$ Q ist nicht in Polynomialzeit lösbar.	☐	☒

Aufgabe 100 ★★

VER-AB **Beweisverfahren**

Ihnen fällt folgende Zahlenspielerei auf:

$$8 \cdot 123456789 = 987654312$$

Wenn man die Ziffern von 1 bis 9 fortlaufend aufsteigend hintereinander schreibt und die entstehende Zahl (im Dezimalsystem) mit 8 multipliziert, drehen sich die Ziffern also um, mit Ausnahme der beiden hintersten. Sie denken noch etwas nach, und schließlich stellen Sie fest, dass für verschiedene Basen $b \neq 10$ folgende Verallgemeinerung der obigen Formel gilt (unformal geschrieben in b-adischer Darstellung):

$$(b-2) \cdot 123456789\text{ABC}\cdots[b-1] = [b-1]\cdots\text{CBA}987654312$$

Wieder dreht sich also eine aus den fortlaufenden Ziffern 1 bis $[b-1]$ bestehende Zahl bei Multiplikation mit $(b-2)$ komplett um, mit Ausnahme der beiden hintersten Ziffern.

a) Welche Beweisverfahren gibt es, mit denen man prinzipiell die Vermutung beweisen könnte, dass obige Formel für alle Basen $b > 2$ gilt?

Lösung:

Es gibt folgende Möglichkeiten, die Vermutung zu beweisen:

- Formel aufstellen und direkt beweisen.
- Beweis durch Widerspruch.
- Beweis durch vollständige Induktion.
- (Wenn man den Verdacht hat, dass die Vermutung gar nicht stimmt, dann reicht ein Gegenbeispiel, um sie zu widerlegen.)

b) Wählen Sie eines dieser Verfahren und beweisen Sie die Vermutung.

Lösung:

Wir stellen eine Formel auf und beweisen sie direkt. Die Vermutung kann man formulieren als (der Dreher der letzten beiden Ziffern ist dargestellt als "$-b+1$"):

$$
\begin{aligned}
&(b-2)\cdot\sum_{i=1}^{b-1} i\cdot b^{b-i-1} = \sum_{i=1}^{b-1} i\cdot b^{i-1} - b + 1 \\
\Leftrightarrow\ &\sum_{i=1}^{b-1} i\cdot b^{b-i} - 2\cdot\sum_{i=1}^{b-1} i\cdot b^{b-i-1} = \sum_{i=1}^{b-1} i\cdot b^{i-1} - b + 1 \\
\Leftrightarrow\ &2\cdot\underbrace{\sum_{i=1}^{b-1} i\cdot b^{b-i-1}}_{1\cdot b^{b-2}+\ldots+\underline{(b-1)\cdot b^0}} + \underbrace{\sum_{i=1}^{b-1} i\cdot b^{i-1}}_{(b-1)\cdot b^{b-2}+\ldots+\underline{1\cdot b^0}} - \underbrace{\sum_{i=1}^{b-1} i\cdot b^{b-i}}_{\underline{\underline{1\cdot b^{b-1}}}+2\cdot b^{b-2}+\ldots+(b-1)\cdot b^1} = b-1
\end{aligned}
$$

Durch Zusammenfassen der Summen erhält man:

$$
\begin{aligned}
&\Leftrightarrow \underline{2b-1} - \underline{\underline{b^{b-1}}} + \sum_{i=1}^{b-2}(2\cdot(b-i-1)+(i+1)-(b-i))\cdot b^i = b-1 \\
&\Leftrightarrow (b-1)\cdot\sum_{i=1}^{b-2} b^i = b^{b-1} - b
\end{aligned}
$$

Auflösen der geometrischen Reihe:

$$\begin{aligned} &\Leftrightarrow (b-1) \cdot \frac{b^{b-1} - b}{b-1} = b^{b-1} - b \\ &\Leftrightarrow 1 = 1 \end{aligned}$$

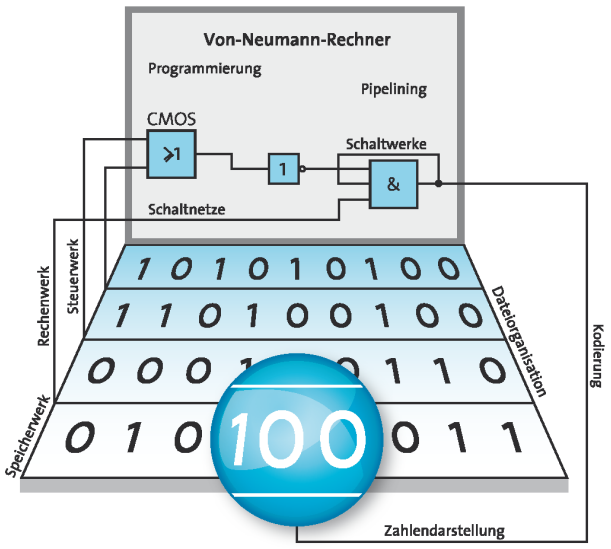
□

Oldenbourg Verlag

Lukas König, Friederike Pfeiffer-Bohnen,
Hartmut Schmeck

100 Übungsaufgaben zu Grundlagen der Informatik

Band II – Technische Informatik



F14

1 Schaltnetze und Schaltwerke

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=357

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 1 ★⋆

CMO-AG **Schaltnetze**

a) Welche Aufgaben haben folgende zwei Konstrukte?

1) Flipflop
2) Register

Lösung:

1) Flipflop: Speicherung eines Bits (Schaltwerk mit mindestens zwei stabilen Zuständen)
   - Einschreiben in den Speicher: Schaltwerk muss definierte Einstellung durch Eingangssignale gestatten
   - Auslesen aus dem Speicher: Speicherinhalt muss in negierter oder nicht-negierter Form an den Schaltwerksausgängen zur Verfügung stehen
2) Register: Speicherung einer Bitfolge, die gewöhnlich als Zahl interpretiert wird
   - Realisiert als Folge synchroner Flipflops mit Hinzunahme eines Takts T

b) Geben Sie an, wie man durch Verwendung von ausschließlich NOR-Gattern die folgenden Gattertypen darstellen kann.

1) AND-Gatter
2) OR-Gatter

Lösung:

1) $\mathrm{AND}(a, b) = \mathrm{NOR}(\mathrm{NOR}(a, a), \mathrm{NOR}(b, b))$
2) $\mathrm{OR}(a, b) = \mathrm{NOR}(\mathrm{NOR}(a, b), \mathrm{NOR}(a, b))$

c) Geben Sie an, wie man das XOR-Gatter ausschließlich durch Nutzung der folgenden Gattertypen darstellen kann.

1) OR-, AND- und NOR-Gatter
2) OR-, AND- und NOT-Gatter

Lösung:

1) $\mathrm{XOR}(a,b) = \mathrm{NOR}(\mathrm{AND}(a,b), \mathrm{NOR}(a,b))$

   oder

   $\mathrm{XOR}(a,b) = \mathrm{NOR}(\underbrace{\mathrm{NOR}(\mathrm{NOR}(a,a), \mathrm{NOR}(b,b))}_{\mathrm{AND}(a,b)}, \mathrm{NOR}(a,b))$

2) $\mathrm{XOR}(a,b) = \mathrm{NOT}(\mathrm{OR}(\mathrm{AND}(a,b), \mathrm{NOT}(\mathrm{OR}(a,b))))$

   oder:

   $\mathrm{XOR}(a,b) = \mathrm{AND}(\mathrm{OR}(a,b), \mathrm{NOT}(\mathrm{AND}(a,b)))$

Aufgabe 2 ⋆⋆

SCH-AA **Schaltnetze**

Ein Volladdierer realisiert auf Schaltkreisebene die Addition von zwei Eingabebits a, b und einem eventuell in früheren Additionen aufgetretenen Übertrag c. Als Ausgabe liefert er ein Summenbit S und ein Übertragsbit C für eventuelle künftige Additionen.

a) Geben Sie eine Wertetabelle für einen Volladdierer an.

Lösung:

Die Wertetabelle enthält die zu addierenden Bits a, b sowie ein Übertragsbit c (carry), und als Funktion davon das Summenbit S und das Übertragsbit für die nächste Stelle C.

a	b	c	S	C
0	0	0	0	0
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	1

b) Ermitteln Sie jeweils eine konjunktive Normalform (KNF) für die beiden Ausgänge. Wie weit kann man diese vereinfachen?

Lösung:

Die KNF lautet jeweils:

$$S = (a \vee b \vee c) \wedge (\overline{a} \vee b \vee \overline{c}) \wedge (a \vee \overline{b} \vee \overline{c}) \wedge (\overline{a} \vee \overline{b} \vee c) = a \oplus b \oplus c$$

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

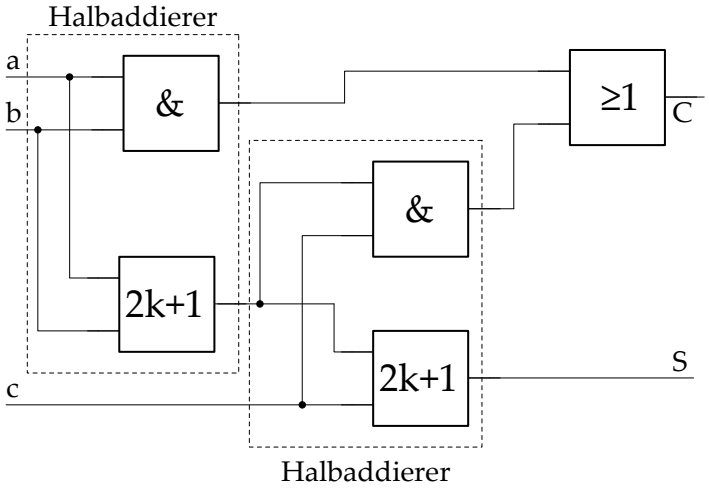
c) Geben Sie eine Realisierung des vereinfachten Schaltnetzes an, indem Sie ausschließlich 2-stellige AND-, OR- und XOR-Gatter verwenden. Legen Sie dabei für C die disjunktive Normalform (DNF) zugrunde und vereinfachen Sie diese.

Lösung:

Wir verwenden für S die Formel aus Teil b) und vereinfachen für C die DNF:

$$\begin{aligned} C &= (\overline{a} \wedge b \wedge c) \vee (a \wedge \overline{b} \wedge c) \vee (a \wedge b \wedge \overline{c}) \vee (a \wedge b \wedge c) \\ &= (\overline{a} \wedge b \wedge c) \vee (a \wedge \overline{b} \wedge c) \vee (a \wedge b) \\ &= [((\overline{a} \wedge b) \vee (a \wedge \overline{b})) \wedge c] \vee (a \wedge b) \\ &= (a \oplus b \wedge c) \vee (a \wedge b) \end{aligned}$$

Es ergibt sich das folgende Schaltnetz:



Aufgabe 3 ★

SCH-AC **Schaltnetze**

Entwerfen Sie ein Schaltnetz, welches zwei vierstellige Dualzahlen

$$x = (d, c, b, a), x' = (d', c', b', a') \in \mathbb{B}^4$$

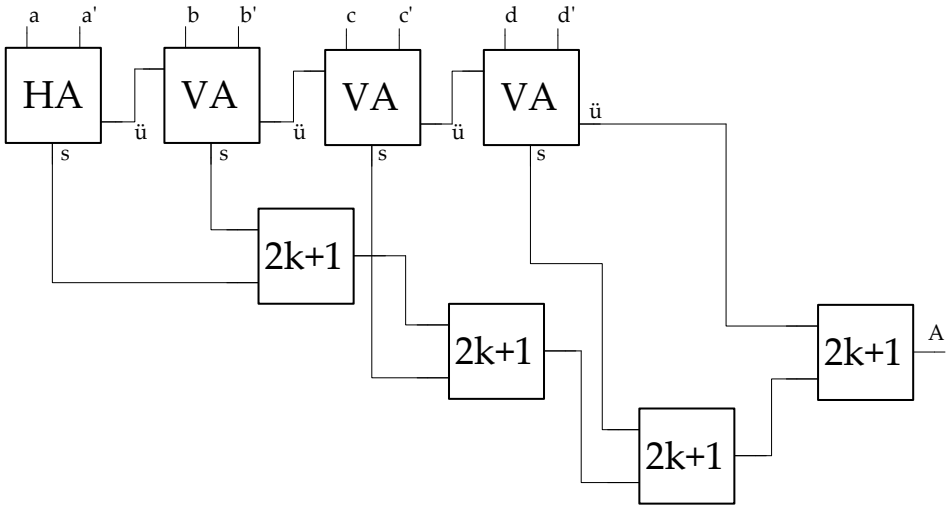
addiert und sich in Abhängigkeit der Einsen im Ergebnis folgendermaßen verhält: bei gerader Anzahl soll am Ausgang A eine 0 anliegen, bei ungerader Anzahl eine 1.
Hinweise:

- Für den Zahlenwert der Bitstrings gilt:

$$\text{wert}(x) = d \cdot 2^3 + c \cdot 2^2 + b \cdot 2^1 + a \cdot 2^0,$$
$$\text{wert}(x') = d' \cdot 2^3 + c' \cdot 2^2 + b' \cdot 2^1 + a' \cdot 2^0$$

- Benutzen Sie als Bausteine die üblichen Gatter AND, OR, NOT, XOR, ... sowie den Halbaddierer HA und den Volladdierer VA.
- Beachten Sie, dass auch der letzte Übertrag zum Ergebnis gehört.

Lösung:



Aufgabe 4 ★⋆

SCH-AB **Schaltnetze**

Eine Lampe wird von drei Schaltern s_1, s_2, und s_3 an- oder ausgeschaltet. Die Lampe soll leuchten, wenn

- s_1 aus, s_2 an, s_3 an, oder
- s_1 an, s_2 aus, s_3 an, oder
- s_1 an, s_2 an, s_3 an ist.

In allen anderen Fällen soll die Lampe nicht leuchten.

a) Stellen Sie eine Wahrheitstabelle auf, die beschreibt, wie sich der Zustand der Lampe L als Funktion $\mathbb{B}^3 \to \mathbb{B}$ von s_1, s_2, s_3 verhält. Dabei soll der Zustand "aus" mit 0 und "an" mit 1 kodiert werden.

Lösung:

Wahrheitstabelle:

s_1	s_2	s_3	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

b) Geben Sie die Schaltfunktion von L in disjunktiver Normalform (DNF) an.

Lösung:

$L(s_1, s_2, s_3) = (\overline{s_1} \wedge s_2 \wedge s_3) \vee (s_1 \wedge \overline{s_2} \wedge s_3) \vee (s_1 \wedge s_2 \wedge s_3)$

c) Geben Sie die Schaltfunktion von L in konjunktiver Normalform (KNF) an.

Lösung:

$L(s_1, s_2, s_3) = (s_1 \vee s_2 \vee s_3) \wedge (s_1 \vee s_2 \vee \overline{s_3}) \wedge (s_1 \vee \overline{s_2} \vee s_3) \wedge (\overline{s_1} \vee s_2 \vee s_3) \wedge (\overline{s_1} \vee \overline{s_2} \vee s_3)$

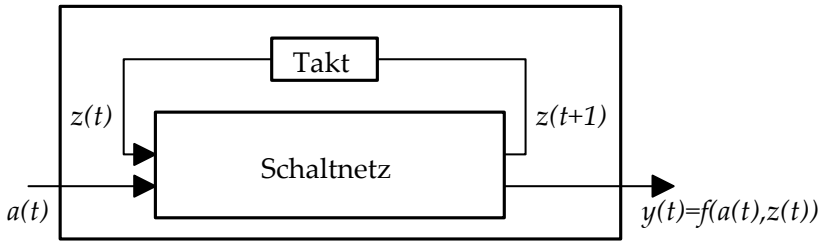
Aufgabe 5 ★

SCH-AE **Schaltwerke**

a) Skizzieren Sie den prinzipiellen Aufbau eines getakteten Schaltwerkes.

Lösung:

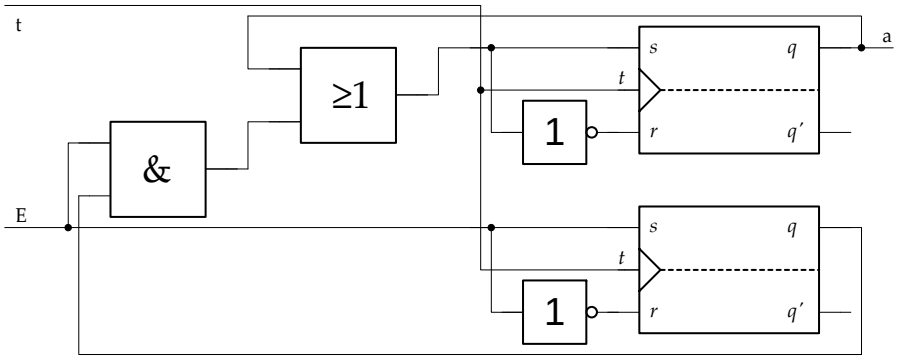
Ein getaktetes Schaltwerk berechnet eine Funktion $y(t) = f(a(t), z(t))$ aus dem zum Zeitpunkt t am Eingang anliegenden Signal $a(t)$ und dem im vorherigen Takt im zugehörigen Schaltnetz berechneten Signal $z(t)$.



b) Skizzieren Sie ein Schaltwerk, das genau dann eine 1 am Ausgang anliegen hat, wenn die bisher erfolgte Eingabe zwei aufeinander folgende Einsen enthält, also für Wörter der Sprache $L = \{\{0,1\}^m 11 \{0,1\}^n \mid m,n \in \mathbb{N}_0\}$. Benutzen Sie dafür nur AND-, OR- und NOT-Gatter sowie getaktete RS-Flipflops.

Lösung:

Durch "scharfes Hinsehen" bzw. logisches Nachdenken kommt man auf folgende Lösung:

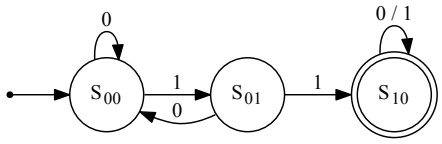


Eine andere Vorgehensweise ist, einen endlichen Automaten für die Sprache L zu erstellen und diesen in ein Schaltwerk umzusetzen.

- Es ergibt sich folgender endlicher Automat:

$$A = (\{0,1\}, \{s_{00}, s_{01}, s_{10}\}, \delta, s_{00}, \{s_{10}\})$$

δ:



- Wir setzen jeden Zustand s_{xy} $(x, y \in \{0,1\})$ des Automaten in die Belegungen $f_1 = x, f_2 = y$ der beiden benötigten Flipflops in der Schaltung um. Wenn wir mit $f_1^\star, f_2^\star$ die Ausgabe des jeweiligen Flipflops im nächsten Takt bezeichnen, ergibt sich folgende Wahrheitstabelle:

E	f_1	f_2	$a = f_1^\star$	$f_2^\star$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	–	–
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	–	–

  Wir setzen $a = f_1^\star$, was bedeutet, dass die Ausgabe der Schaltung genau der Ausgabe des ersten Flipflops entspricht. Es wird also keine weitere Logik am Ausgang benötigt.

- Wir stellen DNFs auf, um die benötigte Logik am Eingang zu ermitteln:

$$\begin{aligned} a = f_1^\star &= E' f_1 f_2' \vee E f_1' f_2 \vee E f_1 f_2' \\ f_2^\star &= E f_1' f_2' \end{aligned}$$

  Die Normalformen können beispielsweise durch Karnaugh-Veitch-Diagramme vereinfacht werden (kein Inhalt der Vorlesung).

- Nun lässt sich das entsprechende Schaltwerk generieren, das ohne Vereinfachung der Formeln allerdings komplizierter ist als das angegebene.

Aufgabe 6 ★

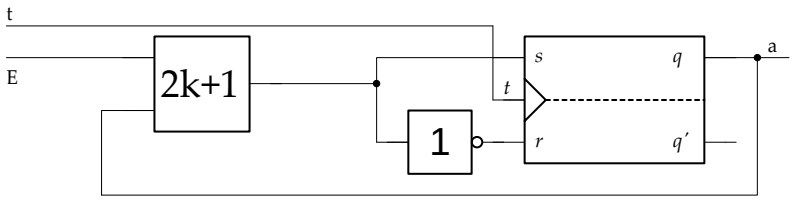
SCH-AF **Schaltwerke**

Skizzieren Sie ein Schaltwerk bei dem genau dann eine 1 am Ausgang a anliegt, wenn die am Eingang E bisher erfolgte Eingabefolge Element folgender Sprache ist:

$$L = \{w \in \{0,1\}^\star \mid |w|_1 \bmod 2 = 1\}$$

Hinweis: Das Schaltwerk erhält pro Takt t ein Signal 0 oder 1, die Länge dieser Eingabe ist unbegrenzt.

Lösung:



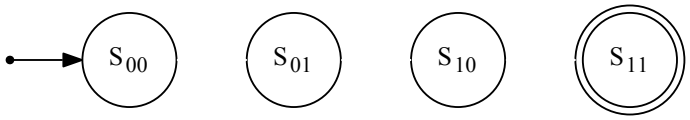
Aufgabe 7 ★★

SCH-AD **Schaltwerke**

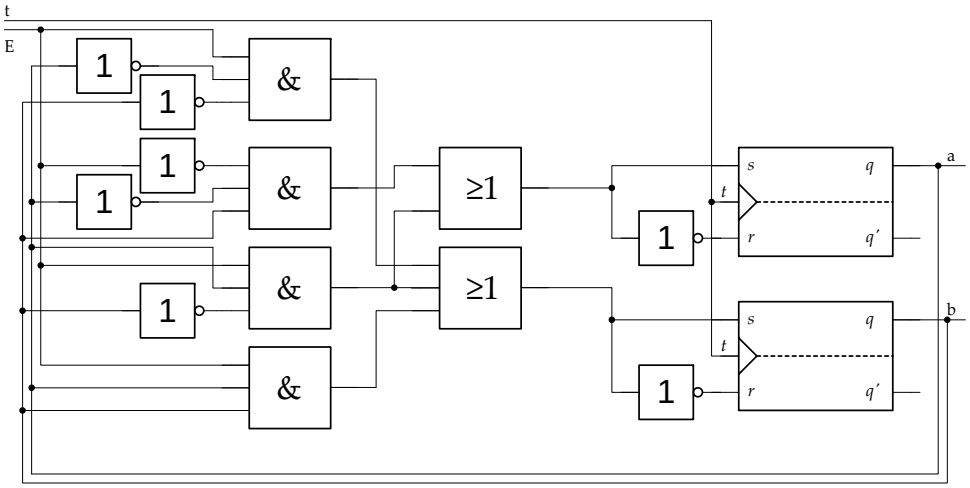
Gegeben seien der unvollständig definierte endliche Automat

$$A = (\{0,1\}, \{s_{00}, s_{01}, s_{10}, s_{11}\}, \delta, s_{00}, \{s_{11}\})$$

δ:



und das folgende Schaltwerk:



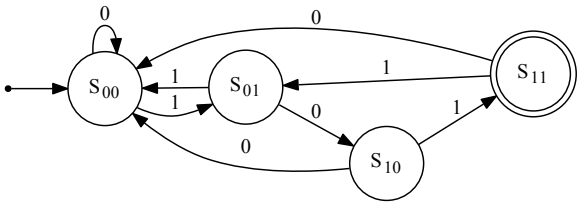
a) Vervollständigen Sie das Zustandsüberführungsdiagramm von A mithilfe des Schaltwerkes, sodass sich bei sukzessiver Abarbeitung eines Wortes $w \in \{0,1\}^\star$ (das über E in das durch t getaktete Schaltwerk eingegeben wird) der Automat genau dann in Zustand s_{ab} befindet, wenn die aktuelle Ausgabe des Schaltwerkes (a, b) ist, wobei $a, b \in \{0, 1\}$.

Lösung:

Zustandsüberführungsdiagramm von A:

$$A = (\{0,1\}, \{s_{00}, s_{01}, s_{10}, s_{11}\}, \delta, s_{00}, \{s_{11}\})$$

δ:



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

Lösung:

$\alpha = (0 + 100 + 11 + 101)^\star 101$

2 Complementary Metal Oxide Semiconductor (CMOS)

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=358

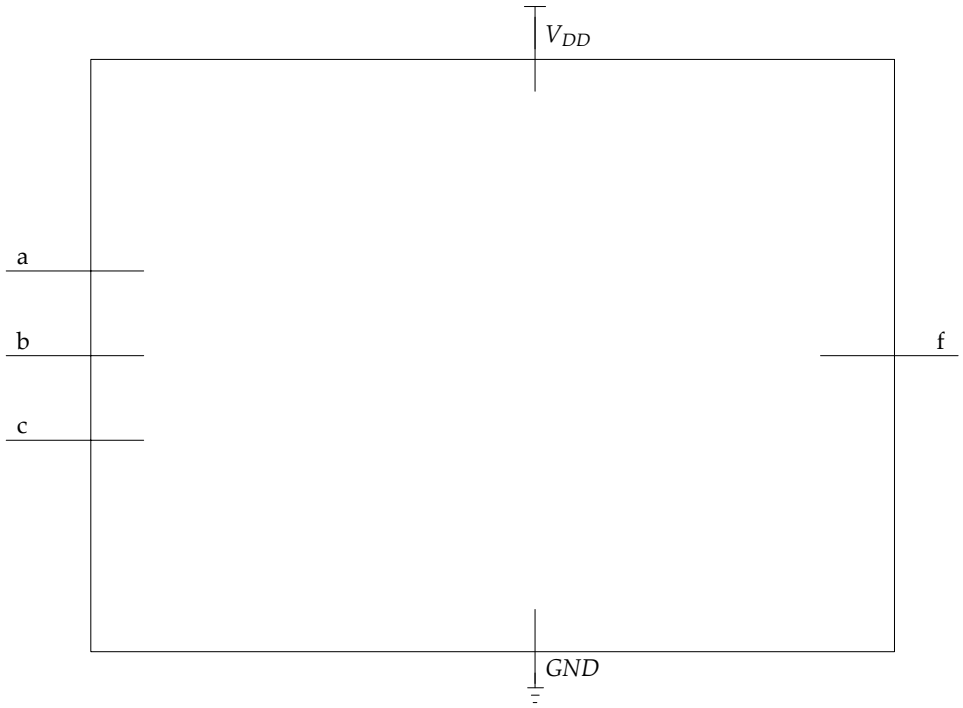
Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 8 ★

CMO-AA **CMOS**

Vervollständigen Sie das folgende CMOS-Schaubild eines 3-stelligen NAND-Gatters, also ein NAND-Gatter mit drei Eingängen. Zeichnen Sie das Gatter auf Transistorebene und nutzen Sie hierzu die Schaltzeichen von NMOS- und PMOS-Transistoren.

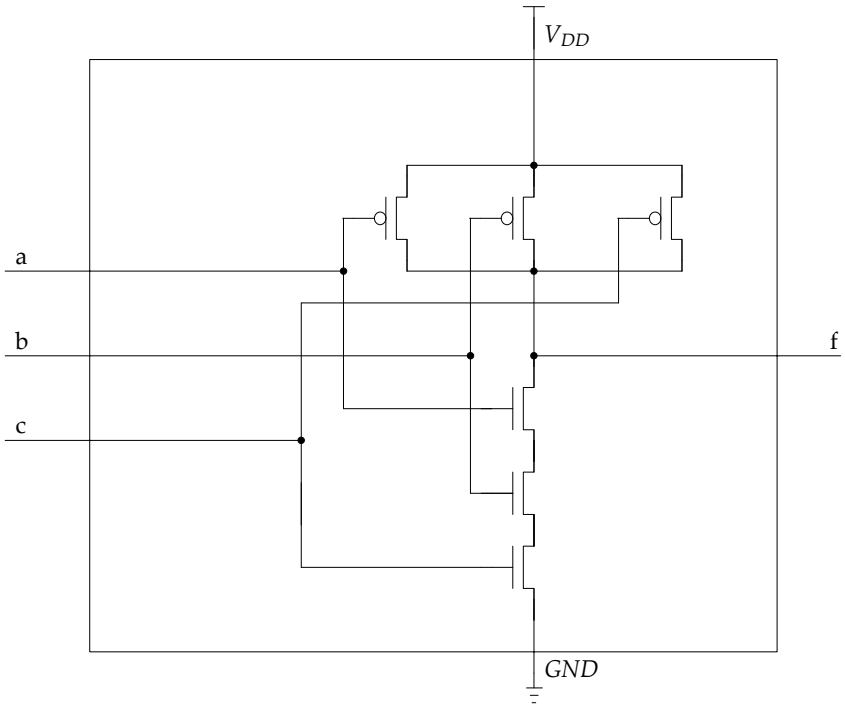


Lösung:

- Boolesche Funktion:

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

- CMOS-Schaubild:



Aufgabe 9 ★

CMO-AF **CMOS**

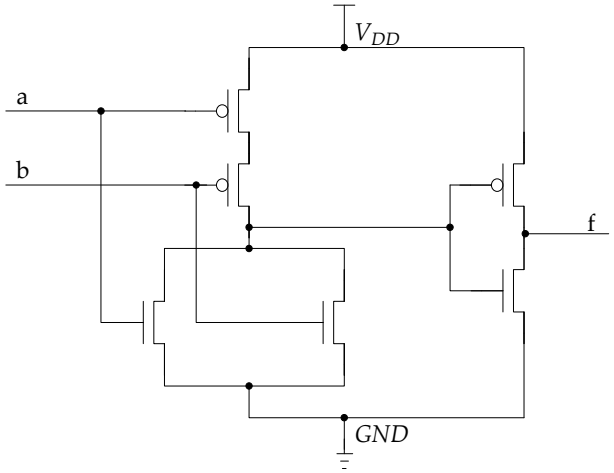
Zeichnen Sie eine CMOS-Schaltung, die ein OR-Gatter realisiert.

Lösung:

- Boolesche Funktion:

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

- CMOS-Schaubild:



Aufgabe 10 ★★

CMO-AC **CMOS**

a) Wofür steht die Abkürzung CMOS?

Lösung:

CMOS steht für Complementary-Metal-Oxide-Semiconductor. CMOS stellt die heutige Standardtechnologie in der Chip-Herstellung dar. Dabei werden PMOS- und NMOS-Transistoren in komplementärer Weise zusammen verbaut.

b) Warum ist PMOS immer an V_{DD} und NMOS immer an GND angeschlossen?

Lösung:

Ein NMOS-Transistor kann die "0" gut übertragen und die "1" schlecht. Ein PMOS-Transistor kann hingegen die "1" gut übertragen jedoch die "0" schlecht.

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

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

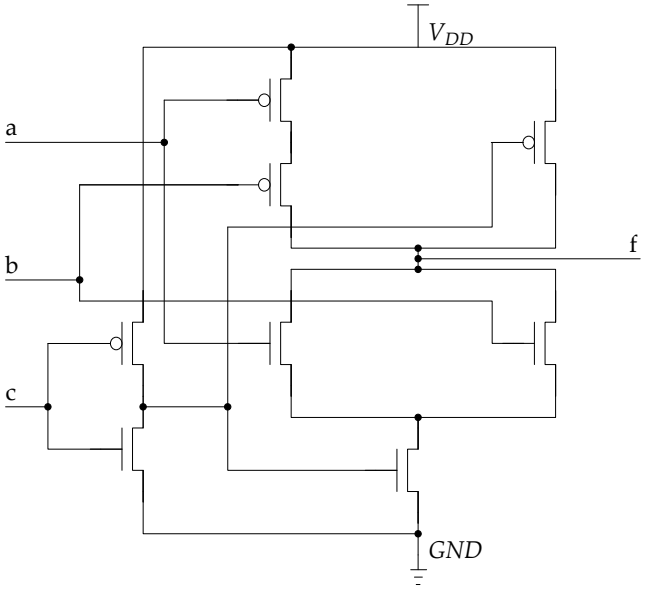
Geben Sie eine CMOS-Schaltung für f an.

Lösung:

- Boolesche Funktion:

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

- CMOS-Schaubild:



Aufgabe 11 ★

CMO-AH **CMOS**

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

$$f(a, b, c) = a \vee b \vee c$$

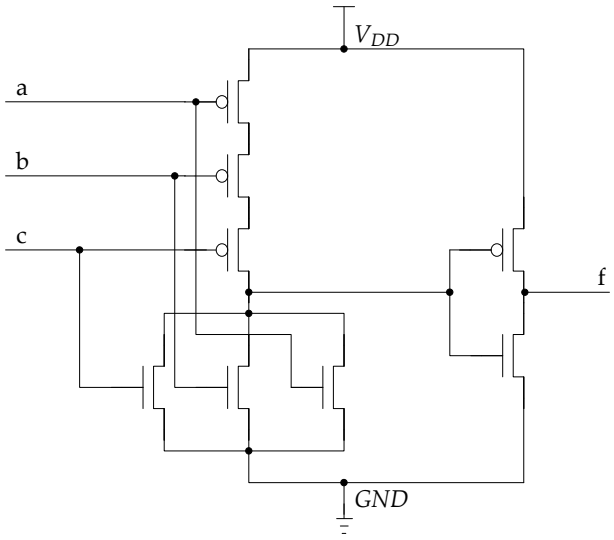
Zeichnen Sie eine CMOS-Schaltung für f.

Lösung:

- Boolesche Funktion:

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

- CMOS-Schaubild:



Aufgabe 12 ★⋆

CMO-AB **CMOS**

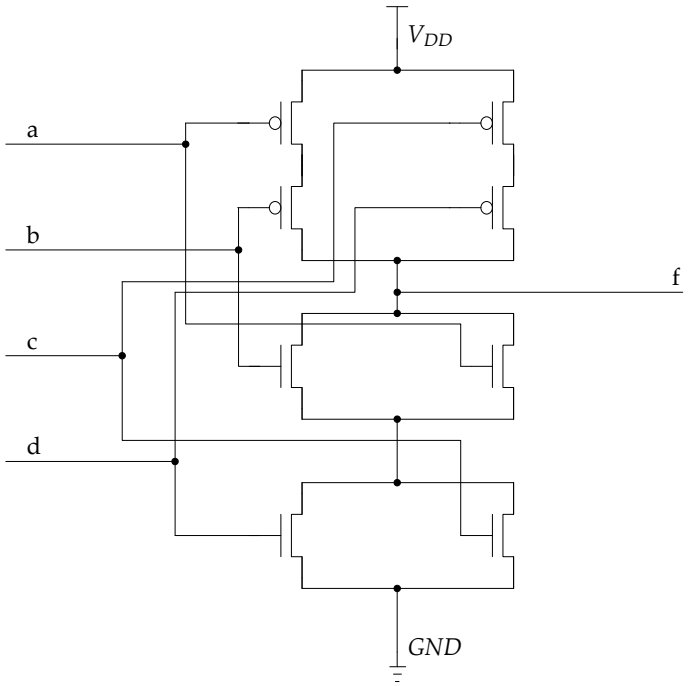
Gegeben sei folgende Schaltfunktion $f : \mathbb{B}^4 \to \mathbb{B}$ mit

$$f(a, b, c, d) = (\neg a \wedge \neg b) \vee (\neg c \wedge \neg d)$$

a) Geben Sie eine CMOS-Schaltung für f an.

Lösung:

- Boolesche Funktion:
$$f = (\neg a \wedge \neg b) \vee (\neg c \wedge \neg d)$$
- CMOS-Schaubild:



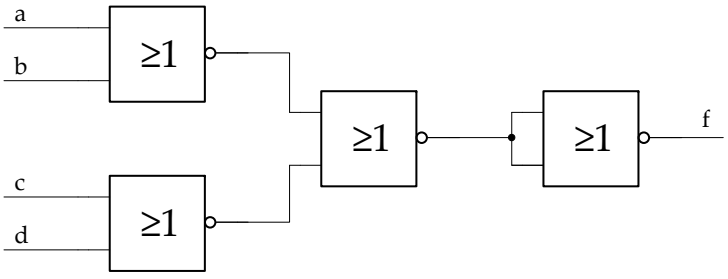
b) Geben Sie ein Schaltnetz für diese Funktion an, indem Sie nur den Baustein NOR verwenden. Benutzen Sie diesen als “Blackbox”.

Lösung:

- Boolesche Funktion:

$$\begin{aligned} f &= (\neg a \wedge \neg b) \vee (\neg c \wedge \neg d) \\ &= \neg(a \vee b) \vee \neg(c \vee d) \\ &= \neg[\neg(\neg(a \vee b) \vee \neg(c \vee d))] \\ &= \neg((a \,\mathrm{NOR}\, b)\,\mathrm{NOR}(c \,\mathrm{NOR}\, d)) \end{aligned}$$

- Schaltnetz:



Aufgabe 13 ★★⋆

CMO-AI **CMOS**

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

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

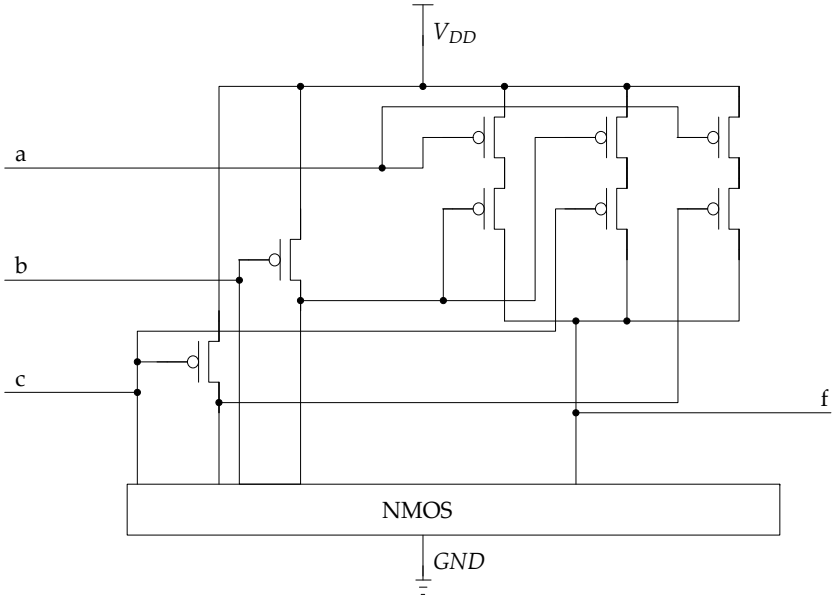
a) Geben Sie eine CMOS-Schaltung für die Funktion f an. Zeichnen Sie dabei nur den PMOS-Bereich auf Transistorebene. Verwenden Sie eine Blackbox für den NMOS-Bereich und stellen Sie durch geeignete Leitungen eine Verbindung zwischen NMOS- und PMOS-Bereich her.

Lösung:

- Boolesche Funktion:

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

- CMOS-Schaubild:



b) Beschreiben Sie in einem Wort, wie Sie den NMOS-Bereich bei einer CMOS-Schaltung im Vergleich zum PMOS-Bereich aufbauen würden.

Lösung:

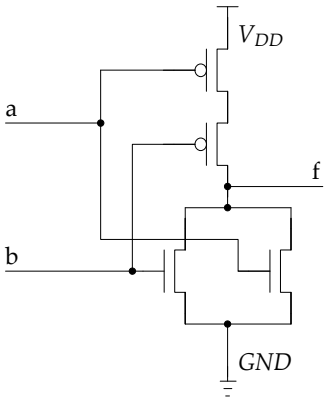
Komplementär

Aufgabe 14 ★

CMO-AE **CMOS**

Welche Funktionen $f : \mathbb{B}^2 \to \mathbb{B} : (a, b) \mapsto f(a, b)$ bzw. $f : \mathbb{B} \to \mathbb{B} : a \mapsto f(a)$ geben folgende CMOS-Schaltungen aus? Geben Sie die Booleschen Funktionen hierfür an.

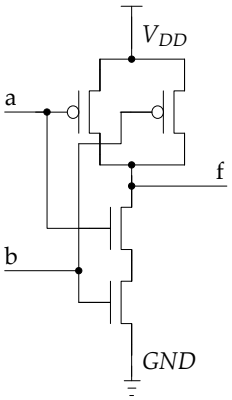
a)



Lösung:

- Schaltung: NOR
- Boolesche Funktion: $f(a, b) = \neg a \wedge \neg b = \neg(a \vee b)$

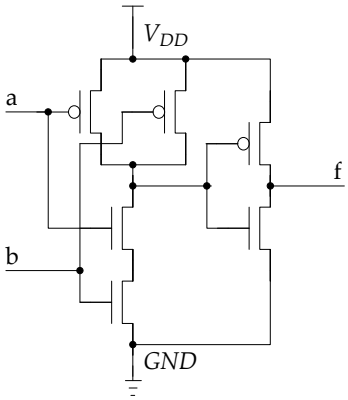
b)



Lösung:

- Schaltung: NAND
- Boolesche Funktion: $f(a, b) = \neg a \vee \neg b = \neg(a \wedge b)$

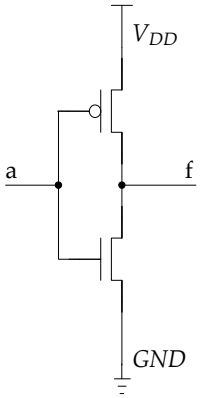
c)



Lösung:

- Schaltung: AND
- Boolesche Funktion: $f(a,b) = \neg(\neg a \vee \neg b) = a \wedge b$

d)



Lösung:

- Schaltung: NOT
- Boolesche Funktion: $f(a) = \neg a$

e) Welches Problem könnte sich ergeben, wenn bei einer Schaltung PMOS an *GND* und NMOS an V_{DD} angeschlossen wären?

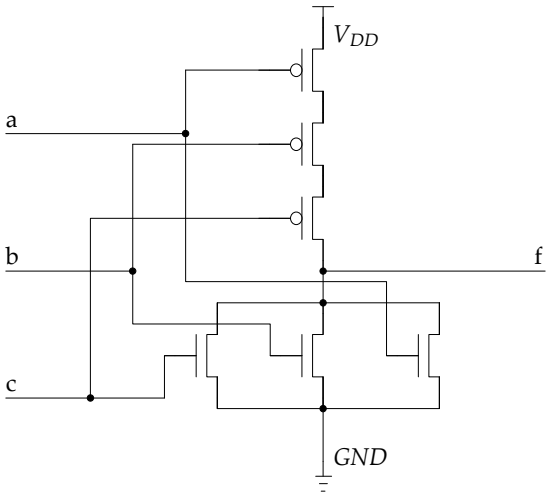
Lösung:

Die Leitfähigkeit könnte unzureichend sein, um die Funktionsweise der Schaltung aufrechtzuerhalten, da NMOS-Transistoren gut die 0 und schlecht die 1 übertragen und PMOS-Transistoren gut die 1 und schlecht die 0.

Aufgabe 15 ⋆

CMO-AD **CMOS**

Gegeben sei folgende CMOS-Schaltung, die eine Boolesche Funktion $f : \mathbb{B}^3 \to \mathbb{B} : (a,b,c) \mapsto f(a,b,c)$ realisiert.



Geben Sie die Funktion f als Booleschen Ausdruck und als Wahrheitstabelle an.

Lösung:

- Wahrheitstabelle:

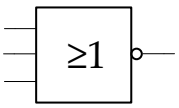
a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

- Boolesche Funktion:

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

Im PMOS-Block kann das Ausgangssignal f nur auf "1" liegen, wenn alle drei PMOS-Transistoren leiten. Die Eingangssignale a, b und c müssen dazu alle auf "0" liegen. Im NMOS-Block liegt das Ausgangssignal f auf "0", wenn mindestens einer der drei NMOS-Transistoren leitet. Für die Eingangssignale a, b und c reicht dazu eine "1". Es leitet jeweils nur einer der Blöcke und der andere sperrt.

Es handelt sich demnach um ein NOR-Gatter mit drei Eingängen.



Aufgabe 16 ★★⋆

CMO-AJ **CMOS**

Gegeben sei eine Boolesche Funktion $f : \mathbb{B}^3 \to \mathbb{B} : (a, b, c) \mapsto f(a, b, c)$, die durch die abgebildete CMOS-Schaltung definiert wird. Geben Sie die Funktion f als Term in Boolescher Algebra an.

Lösung:

- Term nach erster Zusammenführung: $\neg a \wedge \neg b$
- Term nach zweiter Zusammenführung: $\neg(\neg a \wedge \neg b) = a \vee b$
- Term nach dritter Zusammenführung: $\neg(a \vee b) \vee \neg c = \neg a \wedge \neg b \vee \neg c$
  $\to f(a, b, c) = \neg a \wedge \neg b \vee \neg c$

Aufgabe 17 ★⋆

VER-AA **Aussagenlogik, Schaltungen und CMOS**

Lösung:

	Wahr	Falsch
Die Aussagenlogik, die Mengenalgebra und die Schaltalgebra sind Boolesche Algebren. Daher gelten für sie äquivalente Axiome und Umformungsgesetze.	☒	☐
Es kann vorkommen, dass durch zwei verschiedene Verknüpfungsbasen ein und derselben Funktionsmenge unterschiedliche Anzahlen von Funktionen darstellbar sind.	☐	☒
Systeme, die aus ASICs bestehen, können nachträglich nicht umprogrammiert werden, solche aus FPGAs schon.	☒	☐
NMOS-Transistoren liegen in CMOS-Schaltungen tendenziell näher an GND als an V_{DD}.	☒	☐
Durch Dotierung werden Eigenschaften des Siliziumkristalls geändert, sodass er als Transistor verwendet werden kann	☒	☐

3 Binary Decision Diagram

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=359

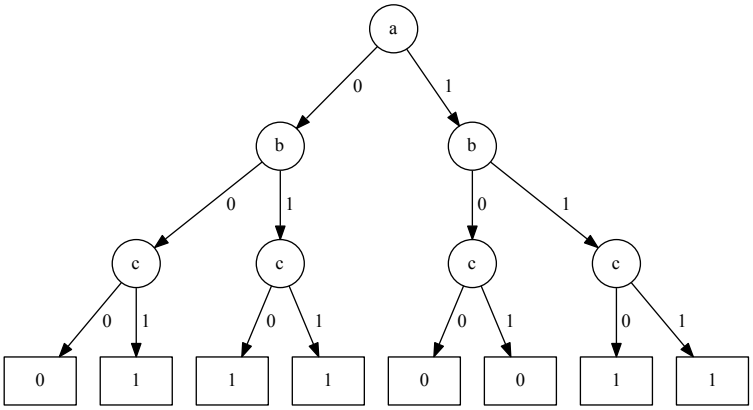
Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 18 ★

BIN-AA **Binary Decision Diagram (BDD)**

Die Boolesche Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ sei durch folgenden Baum gegeben.

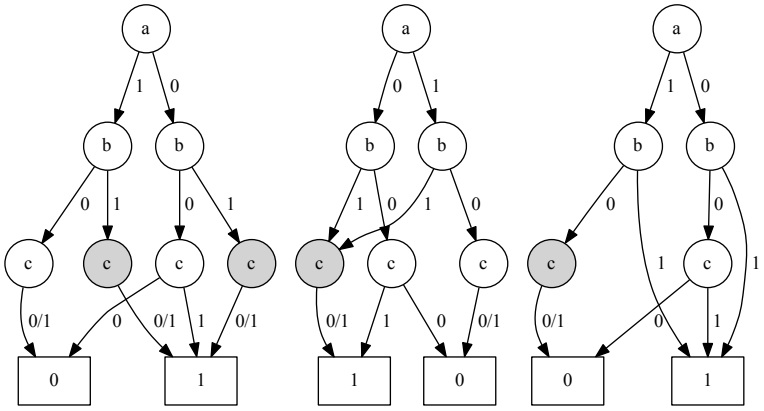


a) Erzeugen Sie das zu f gehörende BDD. Nutzen Sie die im Baum dargestellte Variablenreihenfolge.

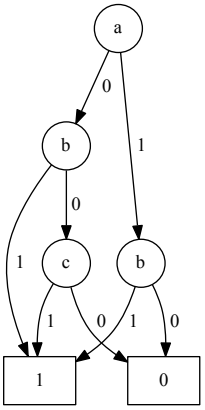
Lösung:

Zuerst werden 0- bzw. 1-Knoten zusammengefasst, danach wird bottom-up nach dem Algorithmus verfahren. In grau werden die Knoten angezeigt, die im nächsten Schritt zusammengefasst bzw. entfernt werden.

- Vereinfachung:



- BDD:



b) Geben Sie die Boolesche Funktion f als Wahrheitstabelle und als Booleschen Ausdruck an.

Lösung:

Wahrheitstabelle:

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

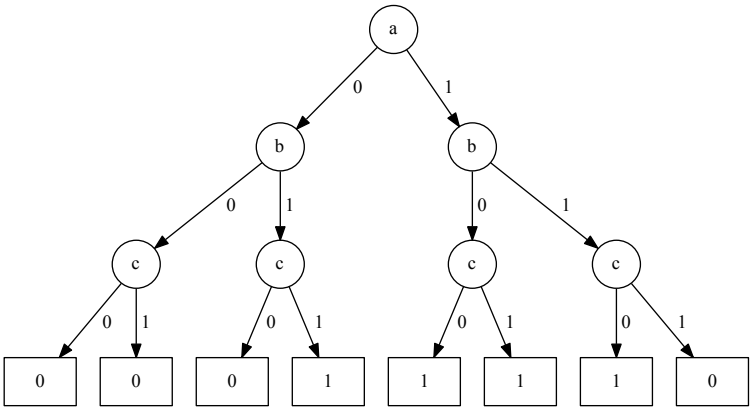
Boolescher Ausdruck (sowohl aus Wahrheitstabelle als auch aus BDD ablesbar):

$$\begin{aligned} f(a,b,c) &= a'b'c + a'bc' + a'bc + abc' + abc \\ &= a'b'c + a'b + ab + bc' + bc \\ &= a'b'c + b \\ &= a'c + b \end{aligned}$$

Aufgabe 19 ★

BIN-AB **Binary Decision Diagram (BDD)**

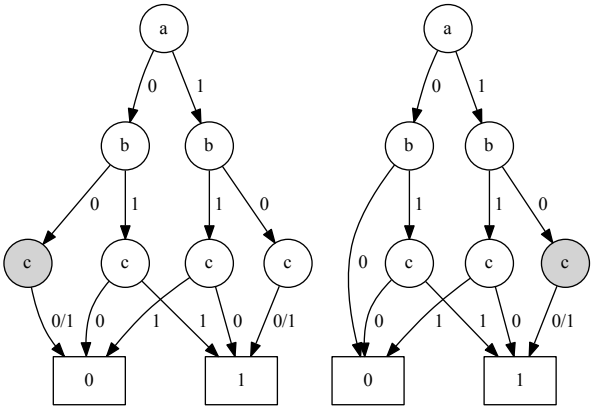
Die Boolesche Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ sei durch folgenden Baum gegeben.



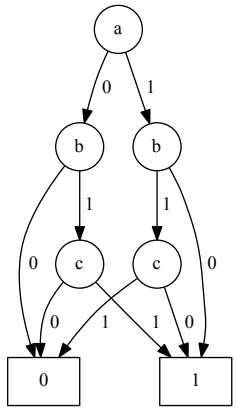
a) Geben Sie das zu f gehörende BDD an. Nutzen Sie die im Graph dargestellte Variablenreihenfolge.

Lösung:

- Vereinfachung:



- BDD:



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

Lösung:

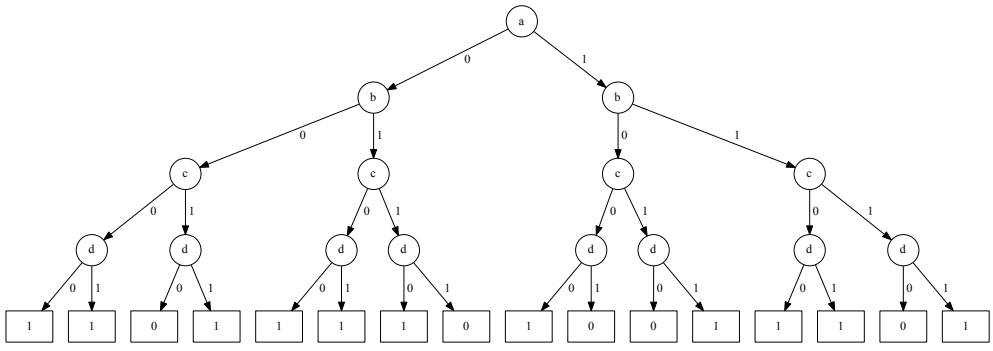
Boolescher Ausdruck:

$$\begin{aligned} f(a,b,c) &= a \cdot b \cdot c' + a \cdot b' + a' \cdot b \cdot c \\ &= a \cdot (bc' + b') + a' \cdot b \cdot c \\ &= a \cdot (c' + b') + a' \cdot b \cdot c \end{aligned}$$

Aufgabe 20 ★★★

BIN-AG **Binary Decision Diagram (BDD)**

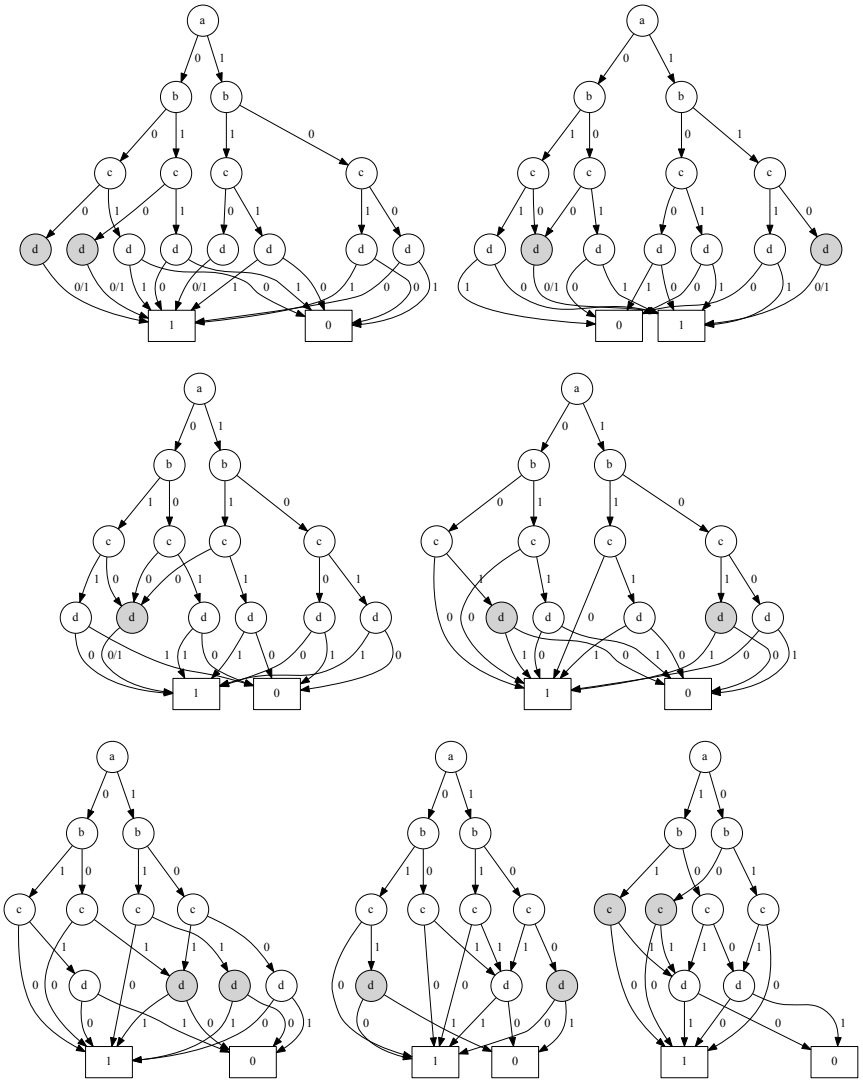
Die Boolesche Funktion $f : \mathbb{B}^4 \to \mathbb{B}$ sei durch den unten abgebildeten Graphen definiert.



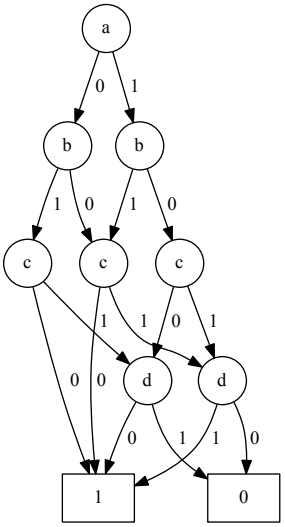
a) Erzeugen Sie das zu f gehörende BDD. Nutzen Sie dazu die im Graph dargestellte Variablenreihenfolge.

Lösung:

- Vereinfachung:



- BDD:



b) Geben Sie die Funktion $f : \mathbb{B}^4 \to \mathbb{B}$ als Booleschen Ausdruck an.

Lösung:

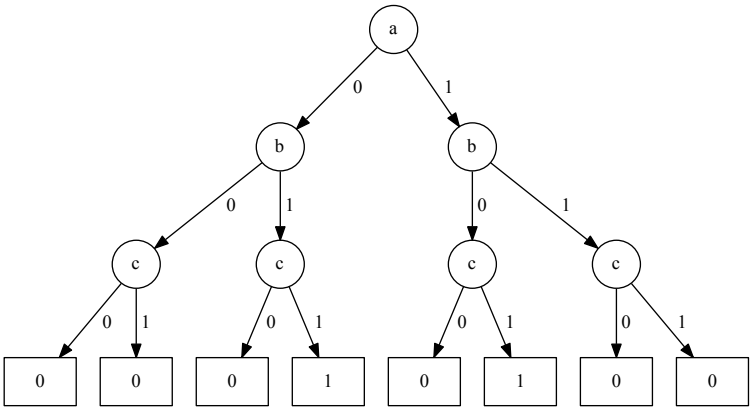
Boolescher Ausdruck als disjunktive Normalform (DNF) aus dem BDD (die Terme in [...] deuten die Herleitung an):

$$
\begin{aligned}
f =& a'bc' + a'bcd' + a'b'c' + a'b'cd + abc' + abcd + ab'c'd' + ab'cd \\
f =& bc'[a'bc' + abc'] + a'bcd' + a'c'[a'b'c' + a'bc'] + a'b'cd + acd[abcd + ab'cd] \\
& + ab'c'd' \\
f =& bc' + a'bcd' + a'c' + a'b'cd + acd + ab'c'd' \\
f =& bc' + a'bcd' + a'c' + b'cd[a'b'cd + acd] + acd + ab'c'd' \\
f =& bc' + a'cd'[a'bcd' + bc'] + a'c' + b'cd + acd + ab'c'd' \\
f =& bc' + a'd'[a'cd' + a'c'] + a'c' + b'cd + acd + ab'c'd' \\
f =& bc' + a'd' + a'c' + b'cd + acd + ab'c'd'
\end{aligned}
$$

Aufgabe 21 ★

BIN-AF **Binary Decision Diagram (BDD)**

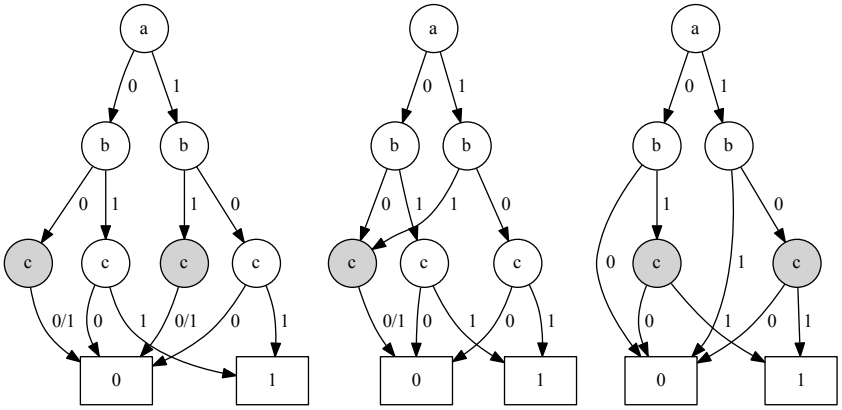
Die Funktion $f : \mathbb{B}^3 \to \mathbb{B}$ sei durch den abgebildeten Graphen gegeben.



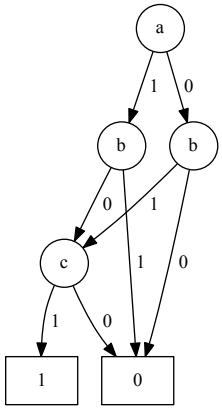
a) Erzeugen Sie das zu der Funktion f gehörende BDD. Nutzen Sie die im Graphen dargestellte Variablenreihenfolge.

Lösung:

- Vereinfachung:



- BDD:



b) Geben Sie die Boolesche Funktion f als Wahrheitstabelle und als Booleschen Ausdruck an.

Lösung:

Wahrheitstabelle:

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

Boolescher Ausdruck:

$$f = ab'c + a'bc = c(a \oplus b)$$

Aufgabe 22 ★⋆

BIN-AE **Binary Decision Diagram (BDD)**

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

$$f(a, b, c) = (a \oplus b \oplus c) + a$$

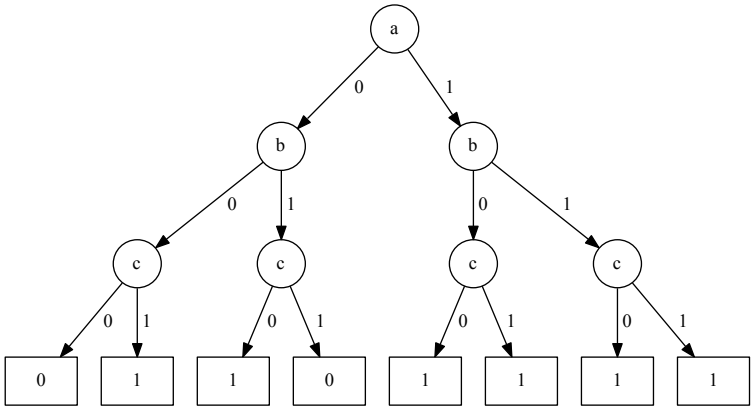
Erzeugen Sie zu f ein BDD mit der Variablenreihenfolge $a \to b \to c$. Geben Sie hierzu zuerst die Wahrheitstabelle für f an.

Lösung:

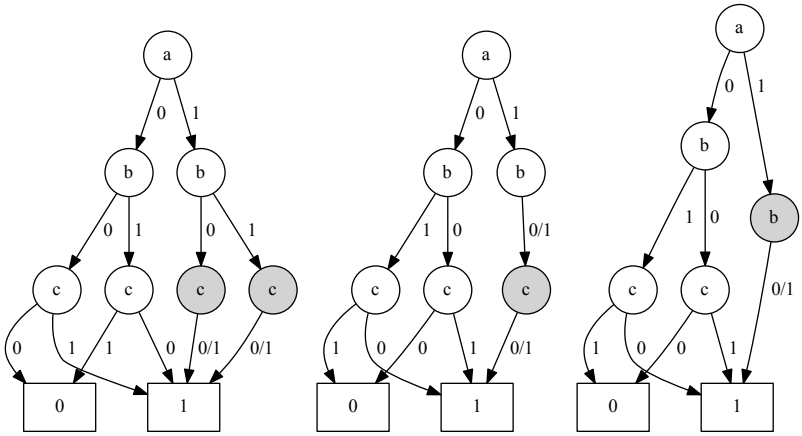
- Wahrheitstabelle:

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

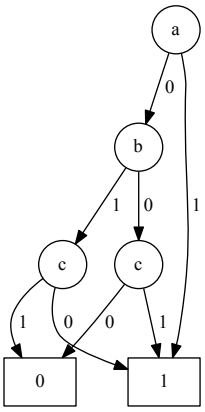
- Entscheidungsbaum:



- Vereinfachung:



- BDD:



Aufgabe 23 ★⋆

BIN-AH **Binary Decision Diagram (BDD)**

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

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

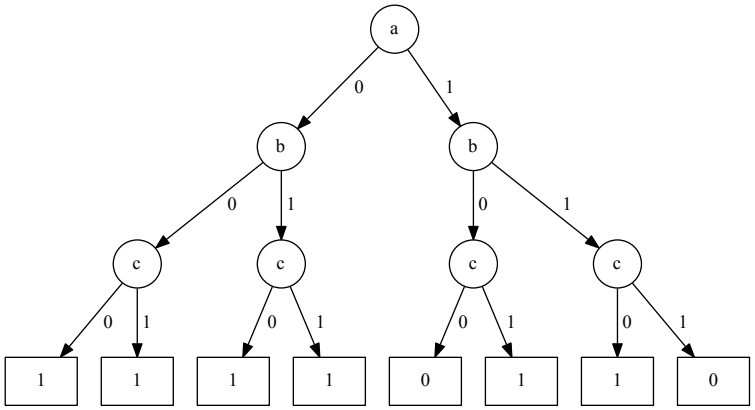
Geben Sie eine Wahrheitstabelle zur Funktion f an und erzeugen Sie das zugehörige BDD. Verwenden Sie hierbei die Variablenreihenfolge $a \to b \to c$.

Lösung:

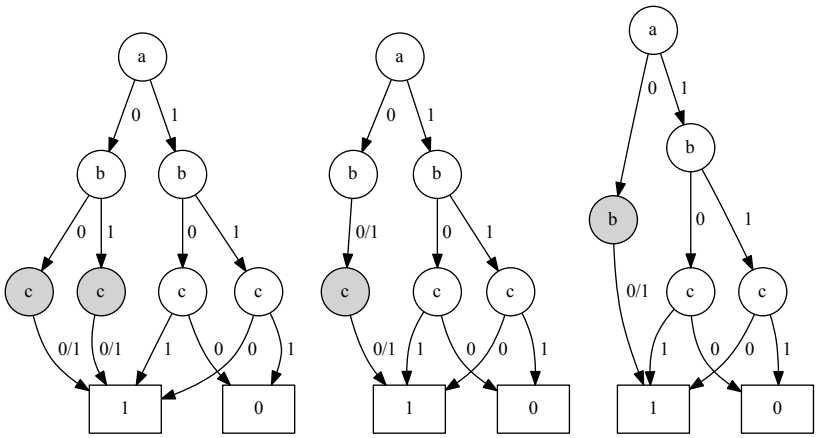
- Wahrheitstabelle:

a	b	c	$a' \cdot b$	$a' \cdot c'$	$a' \cdot b + a' \cdot c'$	$b \oplus c$	$f(a,b,c)$
0	0	0	0	1	1	0	1
0	0	1	0	0	0	1	1
0	1	0	1	1	1	1	1
0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	1
1	1	0	0	0	0	1	1
1	1	1	0	0	0	0	0

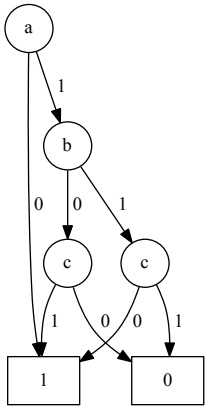
- Entscheidungsbaum:



- Vereinfachung



- BDD:



Aufgabe 24 ★⋆

BIN-AD **Binary Decision Diagram (BDD)**

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

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

Geben Sie das zu f gehörende BDD mit der Variablenreihenfolge $a \to b \to c$ an. Verwenden Sie hierfür den Entwicklungssatz für Booleschen Funktionen:

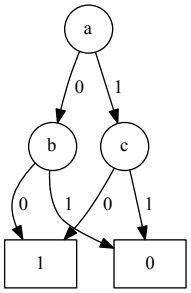
$$f(x_1, x_2, \ldots, x_n) = x_1 \cdot f(1, x_2, \ldots, x_n) + x_1' \cdot f(0, x_2, \ldots, x_n)$$

Lösung:

Nach der Shannon-Formel kann man die Funktion f umschreiben zu:

$$\begin{aligned} f(a, b, c) &= a \cdot f(1, b, c) + a' \cdot f(0, b, c) \\ &= a \cdot (b + c') \cdot (b' + c') + a' \cdot b' \cdot (b' + c') \\ &= a \cdot c' + a' \cdot b' \end{aligned}$$

Damit kommt man zu folgendem BDD:



4 Fehlererkennung und -korrektur, häufigkeitsabhängige Kodierung und Verschlüsselung

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=361

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 25 ⋆

KOD-AB **Fehlerkennbarkeit, Fehlerkorrigierbarkeit**

a) Erklären Sie den Begriff Hammingabstand.

Lösung:

Der Hammingabstand gibt die Anzahl der Stellen an, an denen sich zwei Wörter unterscheiden.

b) Bestimmten Sie den Hammingabstand der Wörter:

- $H\ u\ f\ f\ m\ a\ n$
- $H\ o\ f\ m\ a\ n\ n$

Lösung:

Hammingabstand:

$$h(Huffman, Hofmann) = 4$$

c) Erklären Sie die Begriffe k-Fehlererkennbarkeit und k-Fehlerkorrigierbarkeit für $k \in \mathbb{N}$.

Lösung:

- Fehlererkennbarkeit: Trotz Verfälschung eines Codewortes an bis zu k Stellen kann kein anderes Codewort entstehen: $h_c \geq k + 1$.
- Fehlerkorrigierbarkeit: Trotz Verfälschung eines Codewortes an bis zu k Stellen kann das ursprüngliche Codewort eindeutig ermittelt werden: $h_c \geq 2k + 1$.

d) Wie kann man die sichere Übertragung eines Codes (im Sinne von Teilaufgabe c) erhöhen?

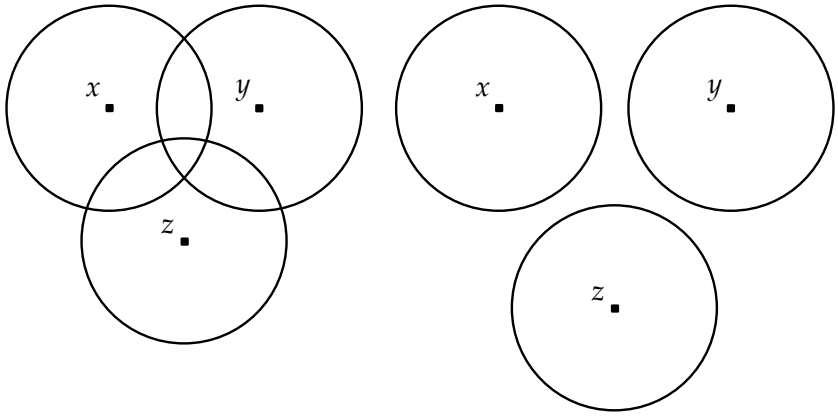
Lösung:

Die sichere Übertragung eines Codes kann beispielsweise durch Anhängen eines Paritätsbits (Prüfbit; XOR-Verknüpfung der Bits im Wort) erhöht werden. Dieses gibt an, ob eine gerade/ungerade Anzahl an Einsen oder Nullen im Wort vorkommt.

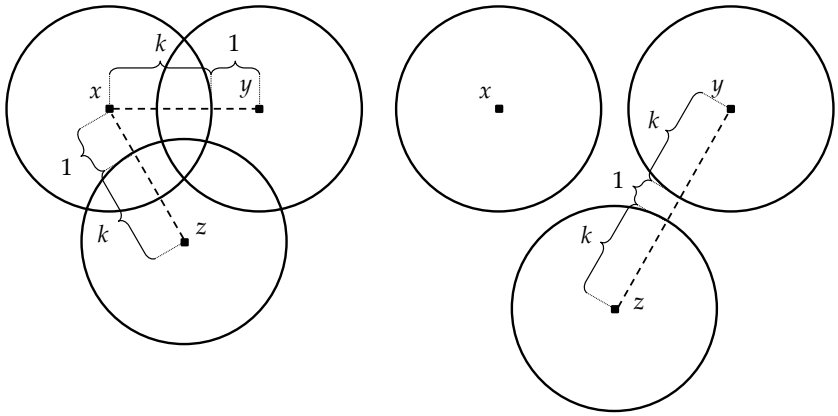
Aufgabe 26 ⋆⋆

KOD-AL **Fehlerkennbarkeit, Fehlerkorrigierbarkeit**

Gegeben seien zwei Darstellungen der Codewörter $x, y, z \in \{0,1\}^\star$ und ihres "Abstands" voneinander. Erklären Sie den Zusammenhang zwischen dem euklidischen Abstand in den Darstellungen und dem Hammingabstand zwischen den Codewörtern. Wie lassen sich aus den Darstellungen die Formeln für die k-Fehlererkennbarkeit bzw. die k-Fehlerkorrigierbarkeit von Codes herleiten?



Lösung:



- Fehlererkennbarkeit: An der linken Abbildung kann man die Fehlererkennbarkeit eines Codes ablesen. Der euklidische Abstand zwischen den Codewörtern x, y und z repräsentiert anschaulich den Hammingabstand zwischen diesen Wörtern. Liegt kein Codewort in einem Kreis mit Radius k um ein anderes Codewort, ergibt sich die Hammingzahl zu $h_c \geq k + 1$. Die Kodierung ist dann mindestens k-fehlererkennbar, da eine Veränderung von höchstens k Bits an einem Codewort nicht ein anderes Codewort ergeben kann. In der Abbildung gilt $h_c = k + 1$ (angedeutet an zwei Beispielen), also ist $k = h_c - 1$.
- Fehlerkorrigierbarkeit: Die rechte Abbildung verdeutlicht die Fehlerkorrigierbarkeit. Alle Kreise mit Radius k um die Codewörter sind hier disjunkt. Die Hammingzahl ergibt

sich zu $h_c \geq 2k + 1$. Die Kodierung ist dann mindestens k-fehlerkorrigierbar, da eine Veränderung von höchstens k Bits an einem Codewort nicht zu einem Wort führen kann, das durch Veränderung von höchstens k Bits aus einem anderen Codewort entstanden ist. In der Abbildung gilt $h_c = 2k + 1$ (angedeutet an einem Beispiel), also ist $k = \left\lfloor \frac{h_c - 1}{2} \right\rfloor$.

Aufgabe 27 ⋆⋆

KOD-AO **Blockkodierungen, Fehlererkennbarkeit, Fehlerkorrigierbarkeit**

Gegeben sei folgende 16-Bit-Kodierung $c : \alpha \to \mathbb{B}^{16}$ mit $\alpha = \{A, B, C, D, E, F, G, H\}$.

$$
\begin{aligned}
c(A) &= 1111\,0000\,0000\,0000 \\
c(B) &= 0000\,1111\,0000\,0000 \\
c(C) &= 0000\,0000\,1111\,0000 \\
c(D) &= 0000\,0000\,0000\,1111 \\
c(E) &= 1100\,1100\,0000\,0000 \\
c(F) &= 0000\,0000\,1100\,1100 \\
c(G) &= 0000\,1100\,1100\,0000 \\
c(H) &= 1100\,0000\,0000\,1100
\end{aligned}
$$

a) Bestimmen Sie die Hammingzahl h_c von c.

Lösung:

Hammingzahl:

$$h_c = \min\{h(x, y) \mid x, y \in \{w \mid \exists v \in \alpha : c(v) = w\}, x \neq y\}$$

mit Hammingabstand $h(x, y)$: Anzahl der Stellen, an denen sich x und y unterscheiden
Es gilt für $U \neq V$:

$$
h(U, V) = \begin{cases}
8, & \text{falls } U, V \in \{A, B, C, D\} \\
 & \text{oder } U, V \in \{C, D, E\} \\
 & \text{oder } U, V \in \{A, F\} \\
 & \text{oder } U, V \in \{A, G\} \\
 & \text{oder } U, V \in \{B, F\} \\
 & \text{oder } U, V \in \{B, H\} \\
 & \text{oder } U, V \in \{C, H\} \\
 & \text{oder } U, V \in \{D, G\} \\
 & \text{oder } U, V \in \{F, E\} \\
 & \text{oder } U, V \in \{G, H\}, \\
4 & \text{sonst}
\end{cases}
$$

Es ergibt sich also die Hammingzahl $h_c = 4$.

b) Die Kodierung c sei k-fehlererkenn- und j-fehlerkorrigierbar. Bestimmen Sie k und j.

Lösung:

- Fehlererkennbarkeit: $k = h_c - 1 = 4 - 1 = 3 \Rightarrow c$ ist 3-fehlererkennbar
- Fehlerkorrigierbarkeit: $j = \left\lfloor \frac{h_c - 1}{2} \right\rfloor = \left\lfloor \frac{4-1}{2} \right\rfloor \Rightarrow c$ ist 1-fehlerkorrigierbar

c) Wäre es möglich, die Hammingzahl durch Anhängen eines Paritätbits zu vergrößern?

Lösung:

Das Anhängen eines Paritätbits würde in diesem Fall keine Vergrößerung der Hammingzahl bewirken. Die Anzahl der Einsen bzw. Nullen in einem Codewort ist immer gerade, sodass das Prüfbit für jedes Codewort den gleichen Wert hätte. Jedoch kann man durch Anhängen eines Paritätsbits immer eine ungerade Anzahl von Fehlern erkennen (aber nicht korrigieren), indem die Parität des übertragenen Wortes überprüft wird.

d) Erfüllt c die Fano-Bedingung?

Lösung:

Die Kodierung erfüllt die Fano-Bedingung, da keines der Codewörter Anfang eines anderen Codewortes ist.

e) Ist die Kodierung c injektiv? Ist die natürliche Fortsetzung von c injektiv?

Lösung:

Die Kodierung c ist injektiv, d. h. jedem Element des Eingabealphabets wird ein anderes Codewort zugeordnet. Da c zusätzlich die Fano-Bedingung erfüllt, folgt die Injektivität der natürlichen Fortsetzung.

Aufgabe 28 ⋆

KOD-AN **Fehlerkennbarkeit, Fehlerkorrigierbarkeit**

Gegeben sei eine Kodierung c mit $c : \alpha \to \mathbb{B}^7$ und $\alpha = \{f, i, l, n, o, s, t, !\}$.

Zeichen	f	i	l	n
Code	0111111	0010011	0100101	0000000

Zeichen	o	s	t	!
Code	1001101	1111001	1010100	1100010

a) Erfüllt c die Fano-Bedingung? Begründen Sie Ihre Antwort kurz.

Lösung:

Ja, die Fano-Bedingung ist erfüllt, da kein Codewort gleich dem Anfang eines anderen Codewortes ist bzw. weil alle Zeichen mit der gleichen Bit-Anzahl kodiert sind und alle 8 Codewörter unterschiedlich sind.

b) Berechnen Sie die Hammingzahl h_c für c. Was bedeutet Ihr Ergebnis für die Fehlererkennbarkeit und die Fehlerkorrigierbarkeit des Codes?

Lösung:

Hammingzahl: $h_c = 3$

- Fehlererkennbarkeit: $h_c \geq k + 1 \Rightarrow$ Der Code ist 2-fehlererkennbar.
- Fehlerkorrigierbarkeit: $h_c \geq 2k + 1 \Rightarrow$ Der Code ist 1-fehlerkorrigierbar.

c) Folgende Zeichenkette wurde übertragen:

0010011010000001111111001101001001111111001

1010100101010010011010100101010010111100010

Was wurde hier übertragen? War die Übertragung fehlerfrei? Wenn nein, welches Zeichen wurde falsch übertragen und was sollte eigentlich übertragen werden?

Lösung:

Übertragene dekodierte Zeichenfolge:
“i” “fehlerhaftes Codewort” “f” “o” “i” “s” “t” “t” “o” “l” “l” “!”

→ Die Übertragung war nicht fehlerfrei; das zweite Zeichen wurde falsch übertragen. Es existiert kein Zeichen, das mit 0100000 kodiert wird. Stattdessen sollte eigentlich “0000000” (“n”) übertragen werden (1-fehlerkorrigierbar).

Aufgabe 29 ⋆

KOD-AP **Kodierung, Fehlererkennbarkeit, Fehlerkorrigierbarkeit**

Es sei folgende 4-Bit-Kodierung $c : \alpha \to \mathbb{B}^4$ mit $\alpha = \{a, b, c, d, e, f\}$ gegeben.

Zeichen	a	b	c	d	e	f
Code	1010	1000	0010	1111	0110	0101

a) Ist die Kodierung c injektiv?

Lösung:

Die Kodierung c ist injektiv (sogar bijektiv), da jedem Element des Ausgangsalphabets ein anderes Codewort zugeordnet wird.

b) Erfüllt c die Fano-Bedingung?

Lösung:

Die Kodierung c erfüllt die Fano-Bedingung, da kein Codewort Präfix eines anderen Codewortes ist.

c) Ist die natürliche Fortsetzung $c^\star$ der Kodierung injektiv?

Lösung:

Die natürliche Fortsetzung $c^\star$ ist injektiv, da die Kodierung c injektiv ist und zudem die Fano-Bedingung erfüllt.

d) Geben Sie die Hammingzahl h_c des Codes und seine Fehlererkennbarkeit sowie Fehlerkorrigierbarkeit an.

Lösung:

- Hammingabstand $h(a, b) = 1$ mit $c(a) = 1010$ und $c(b) = 1000$
- Der Hammingabstand zweier verschiedener Codewörter kann nicht 0 sein, da c injektiv ist.

$\to$ Hammingzahl: $h_c = 1$

- Fehlererkennbarkeit: $k = h_c - 1 = 1 - 1 = 0$: 0-fehlererkennbar
- Fehlerkorrigierbarkeit: $k = \left\lfloor \frac{h_c - 1}{2} \right\rfloor = \left\lfloor \frac{1-1}{2} \right\rfloor = 0$: 0-fehlerkorrigierbar

e) Ändern Sie die Codewörter $c(b)$ und $c(c)$ so, dass die dabei entstandene Kodierung c' 1-fehlererkennbar ist.

Lösung:

Fehlererkennbarkeit: $1 = k = h_{c'} - 1 \rightarrow h_{c'} = 2$

Mit $c(b) = 0000$ und $c(c) = 1001$ gilt $h_{c'} = 2$. Somit ist c' 1-fehlererkennbar.

f) Fügen Sie jedem Codewort der Kodierung c' zwei zusätzliche Bits hinzu, sodass diese 1-fehlerkorrigierbar wird.

Lösung:

Fehlerkorrigierbarkeit: $1 = k = \left\lfloor \frac{h_{c'}-1}{2} \right\rfloor \rightarrow h_{c'} = 3$ (2-fehlererkennbar)

a	1	0	1	0	0	0
b	0	0	0	0	1	1
c	1	0	0	1	0	1
d	1	1	1	1	1	1
e	0	1	1	0	0	1
f	0	1	0	1	0	0

Anmerkung:

Die zusätzlichen Bits können einerseits (wie hier geschehen) durch Ausprobieren hinzugefügt werden. Zudem können die Bits aber auch durch Überprüfung der Parität gewählt werden.

Aufgabe 30 ★⋆

KOD-AC **Fehlererkennbarkeit, Fehlerkorrigierbarkeit**

Betrachten Sie im Folgenden den Aiken-Code $c : \alpha \rightarrow \mathbb{B}^4$ mit $\alpha = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Ziffer	0	1	2	3	4	5	6	7	8	9
Code	0000	0001	0010	0011	0100	1011	1100	1101	1110	1111

a) Ermitteln Sie die Hammingzahl h_c. Wie viel fehlererkennbar ist c, wie viel fehlerkorrigierbar?

Lösung:

Hammingzahl: $h_c = 1 \rightarrow$ 0-fehlererkennbar, 0-fehlerkorrigierbar

b) Im Folgenden soll c um das Prüfbit $p = a_0$ XOR a_1 XOR a_2 XOR a_3 für ein Codewort $w = a_0a_1a_2a_3$ erweitert werden.

1) Geben Sie den erweiterten Code in nachfolgender Tabelle an. Legen Sie hierfür die Bitanordnung $a_0a_1a_2a_3p$ zugrunde.

Lösung:

1) Es ergibt sich folgender erweiterter Code:

$a_0a_1a_2a_3$	$a_0a_1a_2a_3p$
0000	00000
0001	00011
0010	00101
0011	00110
0100	01001
1011	10111
1100	11000
1101	11011
1110	11101
1111	11110

2) Was bewirkt das Anhängen des Prüfbits im Codewort? Ermitteln Sie die Hammingzahl h_c. Wie viele bei der Übertragung eines Codeworts aufgetretene Bit-Fehler können erkannt, wie viele korrigiert werden?

Lösung:

2) Das Anhängen des Prüfbits bewirkt, dass wir immer eine gerade Anzahl Einsen im Codewort haben. Die Hammingzahl h_c erhöht sich zudem um 1.

Hammingzahl: $h_c = 2 \rightarrow$ 1-fehlererkennbar, 0-fehlerkorrigierbar

c) Der Aiken-Code c soll nun für ein Codewort $w = a_0a_1a_2a_3$ um folgende Prüfbits erweitert werden:

$$p_1 = a_0 \text{ XOR } a_1 \text{ XOR } a_2$$
$$p_2 = a_0 \text{ XOR } a_2 \text{ XOR } a_3$$
$$p_3 = a_0 \text{ XOR } a_1 \text{ XOR } a_3$$
$$p_4 = a_1 \text{ XOR } a_2 \text{ XOR } a_3$$

1) Geben Sie den erweiterten Code in nachfolgender Tabelle an. Legen Sie hierfür die Bitanordnung $a_0a_1a_2a_3p_1p_2p_3p_4$ zugrunde.

Lösung:

1) Es ergibt sich folgender erweiterter Code:

$a_0a_1a_2a_3$	$a_0a_1a_2a_3p_1p_2p_3p_4$
0000	00000000
0001	00010111
0010	00101101
0011	00111010
0100	01001011
1011	10110100
1100	11000101
1101	11010010
1110	11101000
1111	11111111

2) Was bewirkt das Anhängen des Prüfbits im Codewort? Ermitteln Sie die Hammingzahl h_c. Wie viele bei der Übertragung eines Codeworts aufgetretene Bit-Fehler können erkannt, wie viele korrigiert werden?
Lösung:

2) Das Anhängen des Prüfbits untersucht jeweils, ob ungerade oder gerade Anzahl Einsen in den Teilen $a_1a_2a_3$, $a_1a_3a_4$, $a_1a_2a_4$ und $a_2a_3a_4$ des Codewortes vorkommen. Insgesamt ergibt sich dann eine gerade Anzahl Einsen im Codewort.

Hammingzahl: $h_c = 4 \rightarrow$ 3-fehlererkennbar, 1-fehlerkorrigierbar

Aufgabe 31 ⋆⋆

KOD-AM **Huffman-Kodierung**

Entscheiden Sie, ob folgende Aussagen wahr oder falsch sind.

Lösung:

	Wahr	Falsch
Eine Huffman-Kodierung ist immer eindeutig bestimmt.	☐	☒
Bei der Huffman-Kodierung werden Zeichenfolgen stets auf Codewortfolgen optimaler Länge abgebildet.	☒	☐
Für eine eindeutige Erzeugung von Codewörtern muss eine Kodierung die Fano-Bedingung erfüllen.	☐	☒
Für eine eindeutige Dekodierung von Codewortfolgen muss die verwendete Kodierung die Fano-Bedingung erfüllen.	☒	☐
Injektive Kodierungen haben injektive natürliche Fortsetzungen.	☐	☒
Erfüllt eine Kodierung die Fano-Bedingung, so ist die natürliche Fortsetzung injektiv.	☐	☒

Aufgabe 32 ⋆

KOD-AE **Huffman-Kodierung**

Erzeugen Sie anhand der durch die folgende Zeichenkette gegebenen Häufigkeitsverteilung eine Huffman-Kodierung c. Ignorieren Sie dabei auftretende Leerzeichen.

FISCHERS FRITZ FISCHT FRISCHE FISCHE

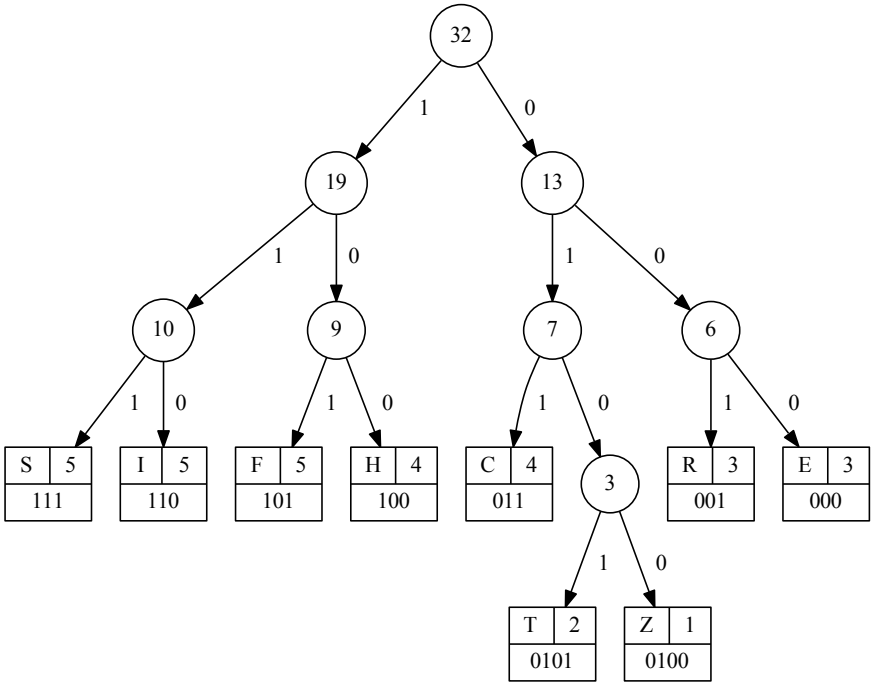
Tragen Sie in die Tabelle die ermittelte Häufigkeit der Zeichen ein und erstellen Sie einen Huffman-Baum. Berechnen Sie die Codelänge $L(c)$ und deren Ersparnis gegenüber der platzsparendsten Kodierung mit einer festen Codelänge.

Lösung:

- Absolute Häufigkeiten:

Zeichen	F	I	S	C	H	R	E	T	Z	Σ
Häufigkeit	5	5	5	4	4	3	3	2	1	32

- Huffman-Baum:



- Kodierung c:

Zeichen	F	I	S	C	H	R	E	T	Z
Code	101	110	111	011	100	001	000	0101	0100

- Codelänge $L(c)$:

$$L(c) = \frac{5}{32}\cdot 3 + \frac{5}{32}\cdot 3 + \frac{5}{32}\cdot 3 + \frac{4}{32}\cdot 3 + \frac{4}{32}\cdot 3 + \frac{3}{32}\cdot 3 + \frac{3}{32}\cdot 3 + \frac{2}{32}\cdot 4 + \frac{1}{32}\cdot 4 = 3{,}09375$$

- Ersparnis: Um 9 Zeichen mit einer Kodierung fester Codelänge darzustellen, werden 4 Bits benötigt.

$$1 - \frac{3{,}09375}{4} \approx 22{,}7\%$$

Aufgabe 33 ★

KOD-AG **Huffman-Kodierung**

Erstellen Sie anhand des folgenden Satzes, dessen Zeichenverteilung repräsentativ sei für einen Text, der kodiert werden soll, eine Huffman-Kodierung (Leerzeichen werden ignoriert).

`ALLE WOLLEN TOLLES INFO`

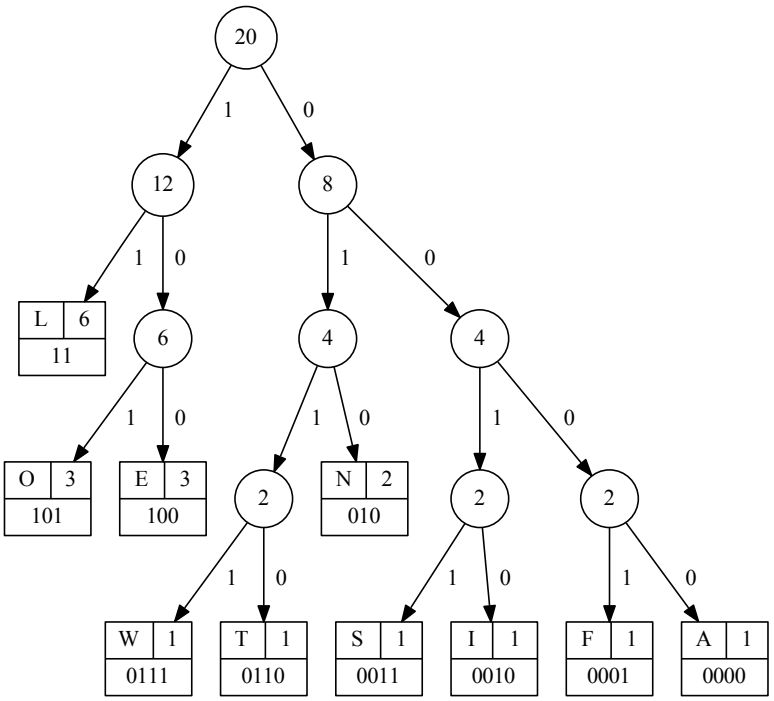
Geben Sie insbesondere einen Huffman-Baum, die Kodierung c der einzelnen Buchstaben und die Codelänge $L(c)$ an.

Lösung:

- Absolute Häufigkeiten:

Zeichen	W	T	S	F	I	A	N	O	E	L	Σ
Häufigkeit	1	1	1	1	1	1	2	3	3	6	20

- Huffman-Baum:



- Huffman-Kodierung c:

Zeichen	W	T	S	F	I	A	N	O	E	L
Code	0111	0110	0011	0001	0010	0000	010	101	100	11

- Codelänge $L(c)$:

$$L(c) = \frac{1}{20}\cdot 4 + \frac{1}{20}\cdot 4 + \frac{1}{20}\cdot 4 + \frac{1}{20}\cdot 4 + \frac{1}{20}\cdot 4 + \frac{1}{20}\cdot 4 + \frac{2}{20}\cdot 3 + \frac{3}{20}\cdot 3 + \frac{3}{20}\cdot 3 + \frac{6}{20}\cdot 2 = \frac{60}{20} = 3$$

Aufgabe 34 ★

KOD-AH **Huffman-Kodierung**

Gegeben sei folgendes Textfragment, das die Zeichen des Alphabets entsprechend ihrer relativen Häufigkeit in einem zu kodierenden Text enthalte.

EGEFUUHFEFEGHUEHHEUGFGFEEKHKUE

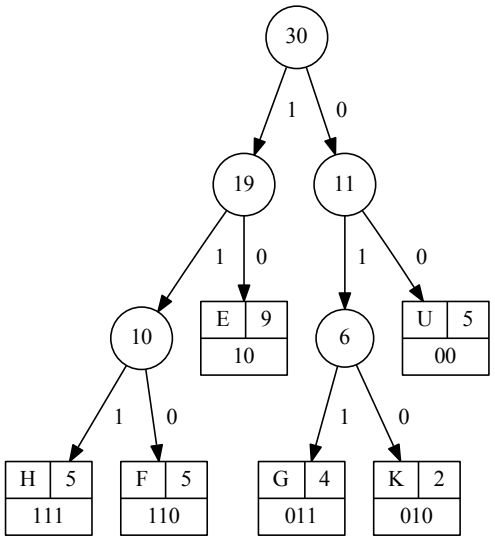
Geben Sie eine Huffman-Kodierung c zu dieser Zeichenverteilung an. Geben Sie hierzu einen Huffman-Baum und die zugehörige Huffman-Kodierung an und berechnen Sie die Codelänge $L(c)$ Ihrer Kodierung.

Lösung:

- Absolute Häufigkeiten:

Zeichen	E	F	H	U	G	K	Σ
Häufigkeit	9	5	5	5	4	2	30

- Huffman-Baum:



- Huffman-Kodierung:

Zeichen	E	F	H	U	G	K
Code	10	110	111	00	011	010

- Codelänge $L(c)$:

$$L(c) = \frac{9}{30} \cdot 2 + \frac{5}{30} \cdot 2 + \frac{5}{30} \cdot 3 + \frac{5}{30} \cdot 3 + \frac{4}{30} \cdot 3 + \frac{2}{30} \cdot 3 = 2\frac{8}{15}$$

Aufgabe 35 ★

KOD-AI **Huffman-Kodierung**

Gegeben sei folgender Text-String, dessen Zeichenverteilung repräsentativ sei für einen zu kodierenden Text ("-" wird auch kodiert).

BAUER-SUCHT-BRAUER

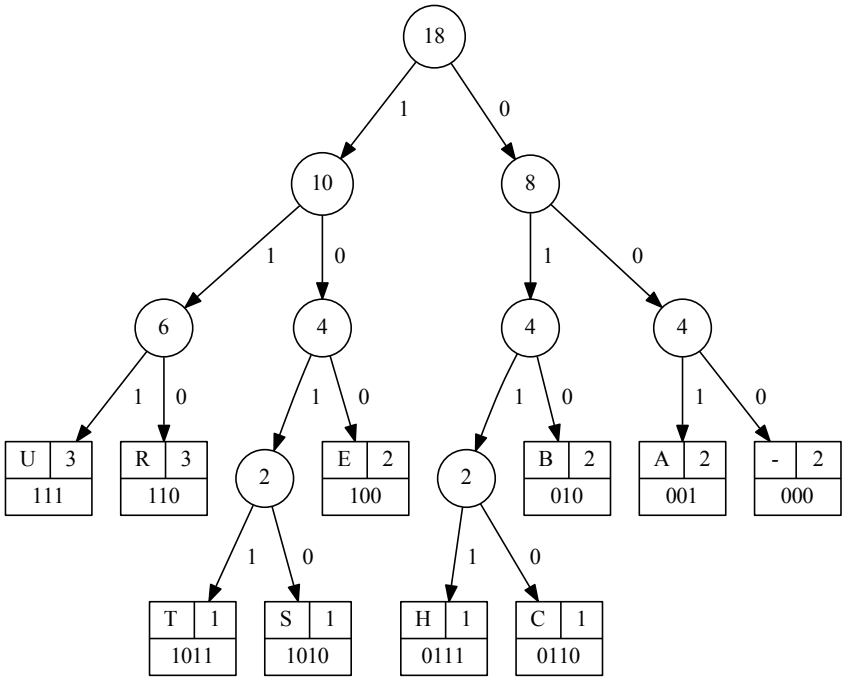
Erzeugen Sie zu der durch den String gegebenen Häufigkeitsverteilung der darin enthaltenen Zeichen eine Huffman-Kodierung c. Geben Sie insbesondere einen Huffman-Baum an.

Lösung:

- Absolute Häufigkeiten:

Zeichen	U	R	B	E	A	$-$	S	C	H	T	Σ
Häufigkeit	3	3	2	2	2	2	1	1	1	1	18

- Huffman-Baum:



- Kodierung c:

Zeichen	U	R	B	E	A	$-$	S	C	H	T
Code	111	110	010	100	001	000	1010	0110	0111	1011

Aufgabe 36 ★

KOD-AF **Huffman-Kodierung**

Die Zeichenverteilung in folgendem Text sei repräsentativ für Text, der zukünftig kodiert werden soll (Leerzeichen werden ignoriert).

`IN ULM UND UM ULM UND UM ULM HERUM`

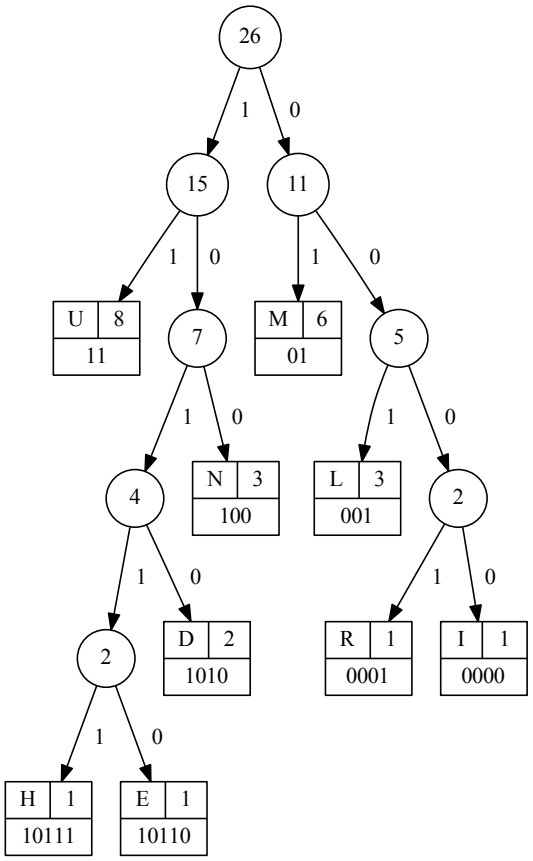
Erzeugen Sie anhand der durch den Text gegebenen Häufigkeitsverteilung eine Huffman-Kodierung c. Wie groß ist die durchschnittliche Länge eines Codewortes?

Lösung:

- Absolute Häufigkeiten:

Zeichen	I	H	E	R	D	N	L	M	U	Σ
Häufigkeit	1	1	1	1	2	3	3	6	8	26

- Huffman-Baum:



- Kodierung c:

Zeichen	U	L	I	H	E	R	D	N	M
Code	11	001	0000	10111	10110	0001	1010	100	01

- Codelänge $L(c)$:

$$L(c) = \frac{8}{26} \cdot 2 + \frac{6}{26} \cdot 2 + \frac{3}{26} \cdot 3 + \frac{3}{26} \cdot 3 + \frac{2}{26} \cdot 4 + \frac{1}{26} \cdot 4 + \frac{1}{26} \cdot 4 + \frac{1}{26} \cdot 5 + \frac{1}{26} \cdot 5 = 2\frac{10}{13}$$

Aufgabe 37 ⋆⋆

KOD-AJ **Huffman-Kodierung**

Gegeben sei folgender Text, dessen Zeichenverteilung repräsentativ sei für einen zu kodierenden Text (Leerzeichen werden nicht kodiert).

`BRAUER BRAUEN BIER`

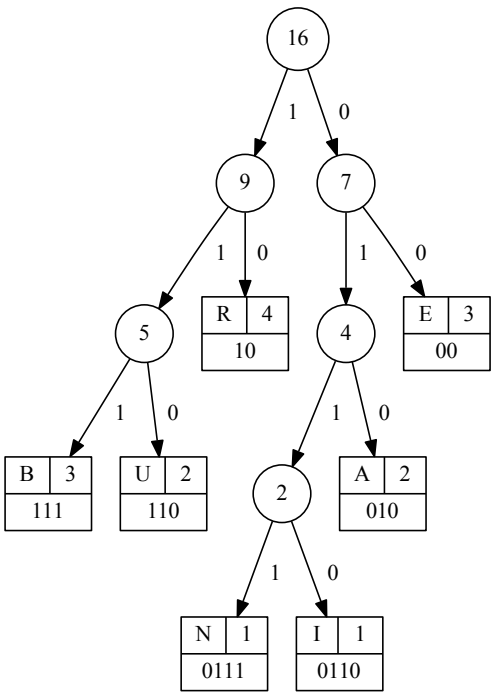
a) Geben Sie die Häufigkeiten der Buchstaben und eine Huffman-Kodierung c für den obigen Text an. Geben Sie insbesondere einen Huffman-Baum an und berechnen Sie die zugehörige Codelänge $L(c)$.

Lösung:

- Absolute Häufigkeiten:

Zeichen	*R*	*B*	*E*	*A*	*U*	*N*	*I*	Σ
Häufigkeiten	4	3	3	2	2	1	1	16

- Huffman-Baum:



- Kodierung c:

Zeichen	R	B	E	A	U	N	I
Code	10	111	00	010	110	0111	0110

- Codelänge $L(c)$:

$$L(c) = \frac{4}{16} \cdot 2 + \frac{3}{16} \cdot 2 + \frac{3}{16} \cdot 3 + \frac{2}{16} \cdot 3 + \frac{2}{16} \cdot 3 + \frac{1}{16} \cdot 4 + \frac{1}{16} \cdot 4 = \frac{43}{16} = 2\frac{11}{16}$$

b) Ist folgende Kodierung c' auch eine Huffman-Kodierung für den oben angegebenen Text? Wenn nicht, geben Sie an, welche Bedingungen nicht erfüllt werden.

Zeichen	R	B	E	A	U	N	I
Code	00	010	11	001	0010	1110	1111

Lösung:

Nein, c' ist keine Huffman-Kodierung für den angegebenen Text:

1) Die Fano-Bedingung ist nicht erfüllt, beispielsweise ist $c'(R) = 00$ ein Präfix von $c'(A) = 001$.
2) Die Kodierung ist auch nicht minimal, da sich die Länge der Codewörter von c und c' nur für U unterscheiden und hier gilt: $|c(U)| < |c'(U)|$.

Aufgabe 38 ★⋆

KOD-AQ **Huffman-Kodierung**

In der folgenden Tabelle sind einige Zeichen mit zugehörigen erwarteten relativen Häufigkeiten für ihr Auftreten in einem zu kodierenden Text gegeben. Weitere Zeichen treten nicht auf.

Zeichen	D	E	I	K	L	O	R
Rel. Häufigkeit	$1/15$	$1/6$	$2/15$	$7/30$	$4/15$	$1/30$	$1/10$

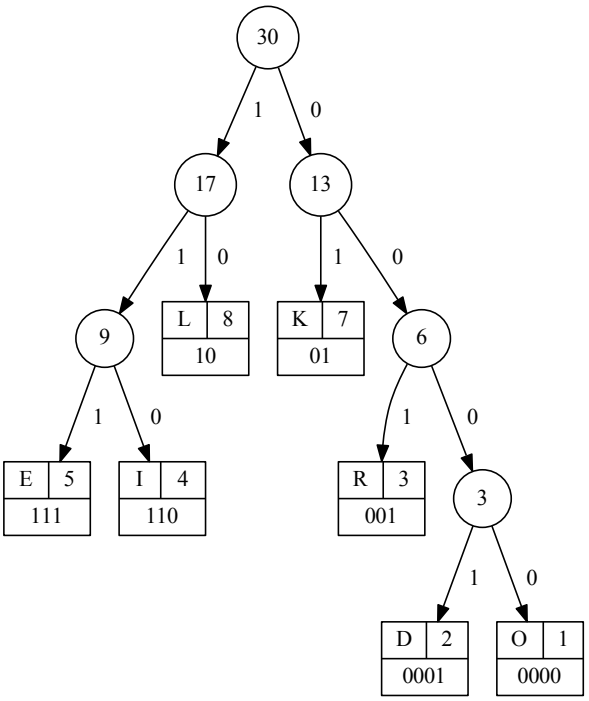
a) Erzeugen Sie eine Huffman-Kodierung c mit der gegebenen Verteilung.

Lösung:

- Absolute Häufigkeiten:

Zeichen	D	E	I	K	L	O	R	Σ
Häufigkeiten	2	5	4	7	8	1	3	30

- Huffman-Baum:



- Kodierung c:

Zeichen	*D*	*E*	*I*	*K*	*L*	*O*	*R*
Code	0001	111	110	01	10	0000	001

b) Dekodieren Sie anhand des ermittelten Huffman-Baums den folgenden Binärstring.

01001000001000000111010111

Lösung:

Dekodierung des Strings:

01 001 0000 01 0000 0001 110 10 111

Das entspricht:

K R O K O D I L E

c) Ist die berechnete minimale Kodierung eindeutig?

Lösung:

Nein, da man zum Beispiel durch Kippen aller Bits ebenfalls eine minimale Kodierung erzeugen kann.

d) Berechnen Sie die Codelänge $L(c)$ dieser Huffman-Kodierung.

Lösung:

Codelänge $L(c)$:

$$L(c) = \frac{1}{30} \cdot 4 + \frac{2}{30} \cdot 4 + \frac{3}{30} \cdot 3 + \frac{4}{30} \cdot 3 + \frac{5}{30} \cdot 3 + \frac{7}{30} \cdot 2 + \frac{8}{30} \cdot 2 = 2{,}6$$

e) Wie viel zusätzlichen Speicherplatz würde eine 3-Bit-Kodierung im Vergleich zu dieser Huffman-Kodierung erfordern?

Lösung:

Zusätzlicher Speicherplatz:

$$\frac{3}{2{,}6} - 1 \approx 15{,}38\%$$

Aufgabe 39 ⋆⋆

KOD-AD **Huffman-Kodierung**

a) Gegeben sei folgende Wahrscheinlichkeitsverteilung für das Auftreten der Zeichen a, b, c, d, e und f in einem zu kodierenden Text.

Zeichen	a	b	c	d	e	f
Wahrscheinlichkeitsverteilung	$^7/_{38}$	$^{10}/_{76}$	$^3/_{76}$	$^9/_{76}$	$^9/_{19}$	$^1/_{19}$

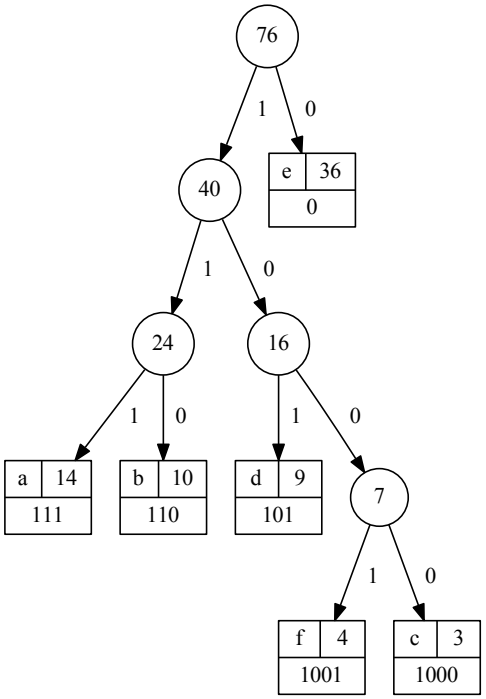
Geben Sie zu dieser Zeichenmenge eine Huffman-Kodierung c an und berechnen Sie die Länge $L(c)$ des Codes. Wie groß wäre die Einsparung gegenüber einer Kodierung, die konstant 4 Bits pro Zeichen benötigt?

Lösung:

- Absolute Häufigkeiten:

Zeichen	a	b	c	d	e	f	Σ
Häufigkeiten	14	10	3	9	36	4	76

- Huffman-Baum:



- Kodierung c:

Zeichen	e	a	b	d	f	c
Code	0	111	110	101	1001	1000

- Codelänge $L(c)$:

$$L(c) = \frac{36}{76} \cdot 1 + \frac{14}{76} \cdot 3 + \frac{10}{76} \cdot 3 + \frac{9}{76} \cdot 3 + \frac{4}{76} \cdot 4 + \frac{3}{76} \cdot 4 = 2\frac{11}{76}$$

- Einsparung gegenüber der 4-Bit-Kodierung:

$$1 - \frac{2^{11}/_{76}}{4} = 46{,}38\%$$

b) Welche zwei Bedingungen muss ein Huffman-Code erfüllen?

Lösung:

- Er muss die Fano-Bedingung erfüllen (kein Codewort darf Präfix eines anderen Codewortes sein).
- Die Codelänge des Codes muss minimal sein.

c) Gegeben seien folgende weitere Codewörter:

g	0111000
h	1010010
i	1110011

Wie lautet die Hammingzahl h_c des Codes?

Lösung:

Codewörter	Hammingabstand
g und h	4
g und i	4
h und i	2

Die Hammingzahl ergibt sich somit zu $h_c = 2$ (minimaler Hammingabstand)

d) Geben Sie an, ob folgende Aussagen richtig oder falsch sind.

Lösung:

	Wahr	Falsch
Die Huffman-Kodierung ist ein Standard-Verfahren zur Datenkompression.	☒	☐
Huffman-Codes sind eindeutig bestimmt.	☐	☒
Die Kodierung aus c) ist 2-fehlererkennbar.	☐	☒
Die Kodierung aus c) ist 1-fehlerkorrigierbar.	☐	☒

Aufgabe 40 ★★⋆

KOD-AK **Vigenère-Verschlüsselung**

Gegeben Sei das Alphabet

$$E = \{\texttt{!}, \texttt{\_}, \texttt{a}, \ldots, \texttt{z}\}$$

Die Symbole seien in der angegebenen Reihenfolge bzw. alphabetisch mit $0, \ldots, 27$ durchnummeriert.

a) Verschlüsseln Sie unter Verwendung des Vigenère-Verschlüsselungsverfahrens mit dem Schlüssel (`_, a, b`), Verschiebungsdistanz (1, 2, 3), die folgende Nachricht:

`a t t a c k _ a t _ d a w n`

Lösung:

b v w b e n a c w a f d x p

b) Sie fangen die folgende Nachricht ab:

y s v m b h _ t h s l z w m f c s f h u d a l h h r j s m s u g c _ o

Sie wissen, dass diese mit einem Schlüssel $(k_0, k_1, k_2, k_3, k_4)$ der Länge 5 nach dem Vigenère-Verschlüsselungsverfahren verschlüsselt wurde. Es ist bekannt, dass ein Teil der Originalnachricht

_ e a s y _

ist. Entschlüsseln Sie den Text.

Lösung:

Da der Schlüssel die Länge 5 hat und der bekannte Teil der Nachricht die Länge 6, wird das Zeichen "_" vorne und hinten mit demselben Buchstaben des Schlüssels chiffriert. Die abgefangene Nachricht enthält nur einen Textteil, der nach dem Prinzip "$\star abcd \star$" aufgebaut ist ($\star \mathrel{\widehat{=}}$ Verschlüsselung von "_" und $abcd \mathrel{\widehat{=}}$ Verschlüsselung von "easy"): der Teil "hudalh". Daraus folgt, dass aus "_" ein "h" wurde und die restlichen Buchstaben können dementsprechend auch zugewiesen werden:

$$k_0 = \text{“(h - \_)”} \bmod 28 = (9-1) \bmod 28 = 8 \mathrel{\widehat{=}} \boxed{\texttt{g}}$$
$$k_1 = \text{“(u - e)”} \bmod 28 = (22-6) \bmod 28 = 16 \mathrel{\widehat{=}} \boxed{\texttt{o}}$$
$$k_2 = \text{“(d - a)”} \bmod 28 = (5-2) \bmod 28 = 3 \mathrel{\widehat{=}} \boxed{\texttt{b}}$$
$$k_3 = \text{“(a - s)”} \bmod 28 = (2-20) \bmod 28 = 10 \mathrel{\widehat{=}} \boxed{\texttt{i}}$$
$$k_4 = \text{“(l - y)”} \bmod 28 = (13-26) \bmod 28 = 15 \mathrel{\widehat{=}} \boxed{\texttt{n}}$$

Man sieht, dass "gobin" ein möglicher Schlüssel ist. Aber man weiß noch nicht, ob dieser Schlüssel die richtige Reihenfolge der Buchstaben wiedergibt, bis man den Schlüssel unter den chiffrierten Text wiederholt aneinanderreiht beginnend bei dem bereits identifizierten Teil "hudalh"):

y s v m b h _ t h s l z w m f c s f **h u d a l h** h r j s m s u g c _ o
b i n g o b i n g o b i n g o b i n **g o b i n** g o b i n g o b i n g o
v i g e n e r e _ c i p h e r _ i s _ **e a s y** _ t o _ d e c r y p t !

Man sieht also dass "bingo" der Schlüssel ist und hat damit die Wahrheit des in der unteren Tabellenzeile entschlüsselten Nachrichtentexts gezeigt.

5 Darstellungen von Ziffern und Zahlen

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=362

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 41 ⋆⋆

KOD-AA **Zahlendarstellung: EBCDIC, BCD, Aiken, Exzess-q**

a) Welche gebräuchlichen n-Bit-Codes zur Darstellung fester Zeichensätze gibt es? Aus wie vielen Bits besteht jeweils die Kodierung eines Zeichens?

Lösung:

Einige der gebräuchlichen n-Bit-Codes lauten:

- ASCII (American Standard Code For Information Interchange): 7-Bit-Code (und 1 Prüfbit) oder 8-Bit-Code
- EBCDIC (Extended Binary Coded Decimal Interchange Code): 8-Bit-Code
- Unicode: 16-Bit pro Zeichen

b) Stellen Sie die Zahlen 5 und 7 als EBCDIC dar.

Lösung:

- $5_{10} = 11110101_{EBCDIC}$
- $7_{10} = 11110111_{EBCDIC}$

c) Wie funktioniert die Tetradenkodierung? Nennen Sie mindestens 3 Vertreter dieser Kodierung.

Lösung:

Die Tetradenkodierung ist eine Blockkodierung, bei der die Ziffern immer durch 4 Bits dargestellt werden.

Vertreter sind

- BCD-Kodierung,
- Exzess-3-Code sowie
- Aiken-Code.

Anmerkung: Der 2-aus-5-Code ist eine Blockkodierung mit 5 Bits.

d) Kodieren Sie folgende Ziffern als BCD-Code, Exzess-3-Code sowie Aiken-Code.

Lösung:

Zahl / Kodierung	BCD	Aiken	Exzess-3
2	0010	0010	0101
6	0110	1100	1001
9	1001	1111	1100

e) Gegeben seien folgende Strings.

1) Kreuzen Sie jeweils an, um welche Codes es sich dabei handeln könnte. Mehrfachnennungen sing möglich.

Lösung:

1)

String	BCD	Aiken	Exzess-3
10001001011010000100100101000011	x		x
01000000000111010011001010110001		x	
01110011010110100100101101100101			x

2) Geben Sie jeweils den Dezimalwert der Strings für die passenden Kodierungen an.

Lösung:

2)

String	BCD	Aiken	Exzess-3
10001001011010000100100101000011	89684943		56351610
01000000000111010011001010110001		40173251	
01110011010110100100101101100101			40271832

Aufgabe 42 ⋆

ZAH-AE **Zahlendarstellung: Exzess-q, BCD, Aiken, Zweikomplement**

a) Im Folgenden sollen 4 Bits zur Darstellung von Zahlen verwendet werden. Geben Sie die Zahl 8 in folgenden Darstellungsarten an:

- Exzess-3-Darstellung
- BCD-Darstellung
- Aiken-Darstellung

Lösung:

- Der Zahlenwert 8 in Exzess-3-Darstellung entspricht der Darstellung der Zahl 8 + 3 = 11 in Dualdarstellung, da eine Verschiebung um 3 vorliegt.

$$11_{10} = 1011_2 \rightarrow 8_{10} = 1011_{Exzess-3}$$

- Der BCD-Code ist eine Blockkodierung mit 4 Bits. Demnach können 4 Bits die Zahlen 0 bis 9 darstellen. Damit ergibt sich:

$$8_{10} = 1000_{BCD}$$

- Der Aiken-Code ist eine Blockkodierung mit 4 Bits. Um eine Zahl d zwischen 5 und 9 darzustellen, wird für den Aiken-Code die Zahl $d + 6$ in Dualdarstellung angegeben (zwischen 0 und 4 wird einfach d in Dualdarstellung angegeben).

$$14_{10} = 1110_2 \rightarrow 8_{10} = 1110_{Aiken}$$

b) Geben Sie an, wie die Zahl 235 in BCD-Darstellung aussieht, wenn 12 Bits zur Verfügung stehen.

Lösung:

Hierfür werden drei 4-Bit-Blöcke des BCD-Codes benötigt:

$$235_{10} = 0010\ 0011\ 0101_{BCD}$$

c) Geben Sie die Zahlen 5 und -7 in Zweikomplement-Darstellung an und addieren Sie die Zahlen direkt in der Zweikomplement-Darstellung.

Lösung:

Zweikomplement-Darstellung:

$$5_{10} = 0101_{2K,4}$$
$$-7_{10} = 1001_{2K,4}$$

Um −7 darzustellen, wird zunächst die 7 in Dualdarstellung dargestellt, man erhält 0111. Werden nun alle Bits gekippt und 1 addiert, so erhält man die obige Darstellung.

Addition:

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \\ + \; 0 & 1 & 0 & 1 \\ & & {\scriptstyle 1} & \\ \hline 1 & 1 & 1 & 0 \end{array}$$

Überprüfung: $5_{10} + -7_{10} = -2_{10} = 1110_{2K,4}$

Aufgabe 43 ★⋆

ZAH-AB **Zahlendarstellung: Einskomplement, Zweikomplement**

a) Welcher Zahlenbereich kann in der Zweikomplement-Darstellung mit $n = 8$ dargestellt werden? Geben Sie die Zahlen 62 und −37 in dieser Darstellung an. Wie berechnet man deren Differenz ohne Umwandlung in Dezimalzahlen?

Lösung:

Darstellung des Zahlenbereiches mit n Bits (unsymmetrisch zum Nullpunkt):

$$A_{2K,n} = \{-2^{n-1}, \ldots, 0, \ldots, 2^{n-1} - 1\}$$

Hier:

$$A_{2K,8} = \{-128, \ldots, 127\}$$

Berechnung der Werte:

$$\begin{aligned} (62)_{10} &= (32 + 16 + 8 + 4 + 2)_{10} \\ &= (0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0)_{10} \\ &= (00111110)_{2K,8} \end{aligned}$$

$$\begin{aligned} (-37)_{10} &= -(32 + 4 + 1)_{10} \\ &= -(0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0)_{10} \\ &= -(00100101)_{2K,8} \\ &= (11011011)_{2K,8} \text{ (“Kippen aller Bits und Addition von 1”)} \end{aligned}$$

Berechnen der Differenz durch Addition des Komplements:

$$(62 - (-37))_{10} = (62 + 37)_{10} = 99_{10}$$

Daher:

$$\begin{array}{rcccccccc} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ + & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & & \\ \hline = & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

b) Was sind die Unterschiede zu a), wenn man statt der Zweikomplement-Darstellung die Einskomplement-Darstellung zugrunde legt?

Lösung:

Darstellung des Zahlenbereiches mit n Bits (symmetrisch zum Nullpunkt, also zwei Darstellungen der Null):

$$A_{1K,n} = \{-2^{n-1} + 1, \ldots, 0, \ldots, 2^{n-1} - 1\}$$

Hier:

$$A_{1K,8} = \{-127, \ldots, 127\}$$

Negative Zahlen erhält man jetzt durch einfaches Kippen der Bits der Darstellung der positiven Zahl: $-37_{10} = (11011010)_{1K,8}$

Die Berechnung der Differenz erfolgt analog zur Zweikomplement-Darstellung.

Aufgabe 44 ★

ZAH-AI **Zahlendarstellung: Zweikomplement, Vorzeichen-Betrag**

a) Geben Sie den Zahlenwert -128 in der Zweikomplement-Darstellung mit 32 Bits an.

Lösung:

Darstellung von 128 als Dualzahl zur Basis 2 mit 32 Bits:

$$128 = 2^7 = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1000\ 0000_2$$

Kippen der Bits:

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0111\ 1111$$

Addition von 1:

$$-128_{10} = 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1000\ 0000_{2K,32}$$

b) Geben Sie den Zahlenwert 8193 in der Zweikomplement-Darstellung mit 32 Bits an.

Lösung:

$8193 = 2^{13} + 2^0 = 0000\,0000\,0000\,0000\,0010\,0000\,0000\,0001_{2K,32}$

c) Geben Sie die Zahlenwerte 33 und −17 in der Vorzeichen-Betrag-Darstellung sowie in der Zweikomplement-Darstellung mit jeweils 16 Bits an.

Lösung:

$33_{10} = 2^5 + 2^0 = 0000\,0000\,0010\,0001_{VZB} = 0000\,0000\,0010\,0001_{2K,16}$

$-17_{10} = -(2^4 + 2^0) = 1000\,0000\,0001\,0001_{VZB} = 1111\,1111\,1110\,1111_{2K,16}$

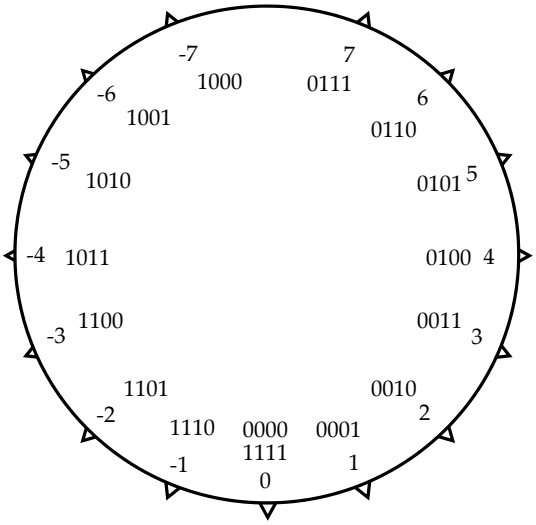
Aufgabe 45 ★

ZAH-AK **Zahlendarstellung: Einskomplement, Zweikomplement**

a) Erläutern Sie, wie die arithmetischen Operationen Negation, Addition und Subtraktion in der Einskomplement-Darstellung durchgeführt werden. Welcher Zahlenbereich kann mit n Bits dargestellt werden?

Lösung:

Darstellbarer Zahlenbereich bei 4 Bits (Einskomplement-Darstellung):



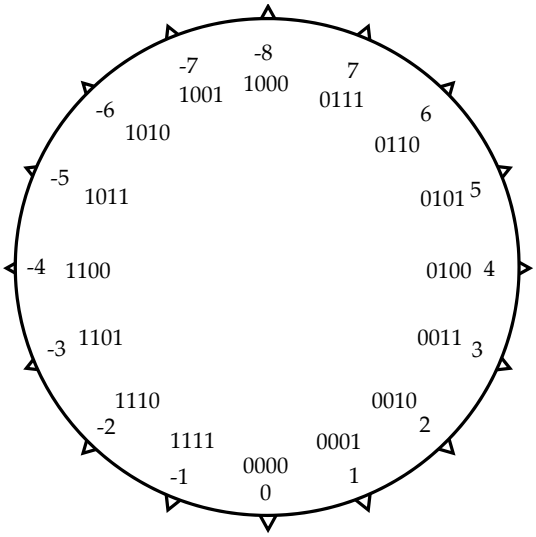
- Negation: Kippen aller Bits.
- Addition: bitweise Addition wie bei Dualzahlen (u. U. ist eine Überlaufbehandlung durch Reduktion mod $2^n - 1$, also durch Addition des Überlaufs notwendig).

- Subtraktion: bitweise Addition der Negation.
- Darstellbarer Bereich: $\{-2^{n-1}+1, \ldots, 0, \ldots, 2^{n-1}-1\}$.
- Weitere Eigenschaften: symmetrisch zum Nullpunkt, es existieren zwei Darstellungen der Null.

b) Erläutern Sie, wie die arithmetischen Operationen aus Aufgabenteil a) in der Zweikomplement-Darstellung durchgeführt werden. Welcher Zahlenbereich ist darstellbar?

Lösung:

Darstellbarer Zahlenbereich bei 4 Bits (Zweikomplement-Darstellung):



- Negation: Kippen aller Bits und duale Addition von 1.
- Addition: bitweise Addition wie bei Dualzahlen (u. U. ist eine Überlaufbehandlung durch Reduktion mod 2^n, also durch Ignorieren des Überlaufs notwendig).
- Subtraktion: bitweise Addition der Negation.
- Darstellbarer Bereich $A = \{-2^{n-1}, \ldots, 0, \ldots, 2^{n-1}-1\}$.
- Weitere Eigenschaften: asymmetrisch zum Nullpunkt, eine Darstellung der Null.

c) Schreiben Sie die Zahl +7 in der Eins- bzw. Zweikomplement-Darstellung mit 4 Bits.

Lösung:

- Einskomplement-Darstellung: $7_{10} = 0111_{1K,4}$
- Zweikomplement-Darstellung: $7_{10} = 0111_{2K,4}$

d) Schreiben Sie die Zahl −7 in der Eins- bzw. Zweikomplement-Darstellung mit 4 Bits.

Lösung:

- Einskomplement-Darstellung: $-7_{10} = 1000_{1K,4}$
- Zweikomplement-Darstellung: $-7_{10} = 1001_{2K,4}$

e) Schreiben Sie die Zahl +12 in der Eins- bzw. Zweikomplement-Darstellung mit 4 Bits.

Lösung:

Mit 4 Bits ist +12 in beiden Darstellungsarten nicht darstellbar, da der darstellbare Zahlenbereich verlassen wird.

Aufgabe 46 ★

ZAH-AD **Zahlendarstellung: IEEE-754**

a) Beschreiben Sie, wie allgemein die Berechnung des Dezimalwerts $G(z)$ einer Zahl z in Gleitpunkt-Darstellung nach IEEE-754 mit einfacher Genauigkeit funktioniert.

Lösung:

Die Zahl sei bitweise gegeben als

$$z \in \mathbb{B}^{32} : z = \underbrace{v}_{\text{Vorzeichenbit}} \quad \underbrace{c_7 \dots c_0}_{\text{Charakteristik } c} \quad \underbrace{m_{-1} \dots m_{-23}}_{\text{Mantisse (ohne impl. 1) } m'}$$

Die Werte für das Vorzeichen V, die Charakteristik C und die Mantisse M werden wie folgt berechnet:

$$V = (-1)^v$$
$$C = \sum_{i=0}^{7} 2^i$$
$$M = 1 + \sum_{i=-23}^{-1} 2^i \cdot m_i$$

Der Exponent E der normierten Gleitpunktzahl wird in Exzess-q-Darstellung gespeichert. Daher benötigt man zur Umrechnung das q, welches sich für eine Charakteristik der Länge $|c| = 8$ wie folgt ergibt:

$$q = 2^{|c|-1} - 1 = 2^7 - 1 = 127$$

Damit ergibt sich der Exponent zu:

$$E = C - q = \sum_{i=0}^{7} 2^i - 127$$

Der Zahlenwert der Gleitpunktzahl ergibt sich dann zu:

$$G(z) = V \cdot 2^E \cdot M = (-1)^v \cdot 2^{\left(\sum_{i=0}^{7} 2^i - 127\right)} \cdot \left(1 + \sum_{i=-23}^{-1} 2^i\right)$$

b) Geben Sie die Zahl $-20^{52}/_{128}$ in IEEE-754-Darstellung an.

Lösung:

Mit der IEEE-754-Darstellung ergibt sich

$$\begin{aligned}
-20^{52}/_{128} &= (-1)^1 \cdot (16 + 4 + {}^{32}/_{128} + {}^{16}/_{128} + {}^{4}/_{128}) \\
&= (-1)^1 \cdot (16 + 4 + {}^{1}/_{4} + {}^{1}/_{8} + {}^{1}/_{32}) \\
&= (-1)^1 \cdot (2^4 + 2^2 + 2^{-2} + 2^{-3} + 2^{-5}) \\
&= \overbrace{(-1)^1}^{V} \cdot \overbrace{2^4}^{2^E} \cdot \overbrace{(1 + 2^{-2} + 2^{-6} + 2^{-7} + 2^{-9})}^{M} \\
\Rightarrow v &= 1 \\
C &= E + q = 4 + 127 = 131_{10} = 10000011_2 \\
c &= 10000011 \\
m' &= 01000110100000000000000 \\
\Rightarrow -20^{52}/_{128} &= 1\ 10000011\ 01000110100000000000000_{GPZ}
\end{aligned}$$

Aufgabe 47 ★

ZAH-AH **Zahlendarstellung: IEEE-754**

a) Geben Sie die Dezimalzahl $-13\,{}^{97}/_{128} = -13,7578125$ in Gleitpunktdarstellung nach IEEE-754 mit einfacher Genauigkeit an.

Lösung:

Gleitpunktzahl gemäß der IEEE-754-Darstellung mit einfacher Genauigkeit bedeutet, dass 32 Bits unterteilt werden in 1 Bit für das Vorzeichen v, 8 Bits für die Charakteristik c und 23 Bits für die Mantisse ohne implizite Eins m'.

$$\begin{aligned} -13{}^{97}/_{128} &= (-1)^1 \cdot (2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-7}) \\ &= (-1)^1 \cdot 2^3 \cdot (1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-10}) \\ &= (-1)^1 \cdot 2^{130-127} \cdot (1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-10}) \end{aligned}$$

Daraus ergibt sich:

$$\begin{aligned} v &= 1 \\ c &= 10000010 \\ m' &= 10111000010000000000000 \\ -13{}^{97}/_{128} &= 1\ 10000010\ 10111000010000000000000_{GPZ} \end{aligned}$$

b) Geben Sie den Dezimalwert $G(x)$ der folgenden Gleitpunktzahl x gemäß IEEE-754 an:

$$x = 1\ 10011011\ 01110010000000000000000$$

Lösung:

Gemäß IEEE-754-Darstellung ergibt sich

$$\begin{aligned} V &= 1 \\ C &= 10011011_2 = 155_{10} \\ E &= 155 - 127 = 28 \\ M &= 1 + 0,01110010000000000000000_2 = {}^{185}/_{128} = 1{,}4453125 \end{aligned}$$

$$G(x) = (-1)^V \cdot 2^E \cdot M = -2^{28} \cdot {}^{185}/_{128} = -387973120$$

Aufgabe 48 ★⋆

ZAH-AA **Zahlendarstellung: IEEE-754**

Gegeben sei eine IEEE-754-ähnliche 8-Bit-Darstellung von Gleitpunktzahlen, bei der ein Binärstring $z \in \mathbb{B}^8$ aufgeteilt ist in ein Bit v, das für das Vorzeichen steht, 3 Bits $c_2c_1c_0$ für die Charakteristik und 4 Bits $m_{-1}m_{-2}m_{-3}m_{-4}$ für die Mantisse:

$$z = v\ c_2c_1c_0\ m_{-1}m_{-2}m_{-3}m_{-4}$$

Der Zahlenwert von z wird nach der für Gleitpunktzahlen üblichen Formel berechnet. Bestimmen Sie die Dezimalwerte folgender Zahlen:

- 0 101 1101
- 1 001 0001
- 1 110 1110

Lösung:

Die Mantisse ohne implizite Eins m' wird durch folgende Formel berechnet:

$$m' = \sum_{i=-4}^{-1} 2^i \cdot m_i$$

Die Charakteristik c wird durch folgende Formel berechnet:

$$c = \sum_{i=0}^{2} 2^i \cdot c_i$$

Der Zahlenwert $G(z)$ einer als Gleitpunktzahl in obiger Darstellung kodierten Zahl z wird folgendermaßen berechnet:

$$G(z) = (-1)^v \cdot 2^{c-q} \cdot (1 + m')$$

Dabei berechnet sich q als:

$$q = 2^{3-1} - 1 = 3$$

Es ergibt sich:

$$\begin{aligned} G(0\ 101\ 1101) &= (-1)^0 \cdot 2^{5-3} \cdot (1 + 2^{-1} + 2^{-2} + 2^{-4}) = {}^{29}/_{4} \\ G(1\ 001\ 0001) &= (-1)^1 \cdot 2^{1-3} \cdot (1 + 2^{-4}) = -{}^{17}/_{64} \\ G(1\ 110\ 1110) &= (-1)^1 \cdot 2^{6-3} \cdot (1 + 2^{-1} + 2^{-2} + 2^{-3}) = -15 \end{aligned}$$

Aufgabe 49 ★★⋆

ZAH-AJ **Zahlendarstellung: Zweikomplement, IEEE-754**

a) Stellen Sie die beiden Werte −12,43 und 27,875 möglichst genau als Festpunktzahlen mit jeweils 16 Vor- und 16 Nachkommabits dar. Nutzen Sie die Zweikomplement-Darstellung.

Lösung:

- Darstellung von -12,43 als Festpunktzahl.

  Die Zahl −12,43 ergibt sich als Zweikomplement der Dualdarstellung von 12,43. Die Vorkommazahl lässt sich wie folgt zerlegen: $12 = 2^3 + 2^2$. Also ist die Binärdarstellung der Vorkommazahl $0^{12}1100$.

  Für die Berechnung der Nachkommabits wenden wir einen Trick an: Wir multiplizieren nur die Nachkommastellen wiederholt mit 2. Ergibt sich dabei eine 1 vor dem Komma, muss auch das entsprechende Bit gesetzt werden. Wenn sich keine 1 ergibt, steht eine 0 an der entsprechenden Bit-Position. Dieses Verfahren wiederholen wir (und ignorieren dabei die Vorkommastellen beim Multiplizieren) so lange, bis wir eine Periodizität der Nachkommabits erkennen oder 16 Nachkommabits berechnet haben.

Stelle	Berechnung	Stelle	Berechnung
0	0,43 · 2 = **0**,86	8	0,08 · 2 = **0**,16
1	0,86 · 2 = **1**,72	9	0,16 · 2 = **0**,32
2	0,72 · 2 = **1**,44	10	0,32 · 2 = **0**,64
3	0,44 · 2 = **0**,88	11	0,64 · 2 = **1**,28
4	0,88 · 2 = **1**,76	12	0,28 · 2 = **0**,56
5	0,76 · 2 = **1**,52	13	0,56 · 2 = **1**,12
6	0,52 · 2 = **1**,04	14	0,12 · 2 = **0**,24
7	0,04 · 2 = **0**,08	15	0,24 · 2 = **0**,48

  Unsere gesuchte Festpunkt-Darstellung von 12,43 lautet also:

  0000 0000 0000 1100 , 0110 1110 0001 0100

  Nun müssen wir noch das Zweikomplement der Zahl berechnen, um die Darstellung von −12,43 zu erhalten. Dazu invertieren wir die Zahl und addieren 1 auf die niederwertigste Stelle:

  1111 1111 1111 0011 , 1001 0001 1110 1100

- Darstellung von 27,875 als Festpunktzahl.

  Die Vorkommazahl lässt sich wie folgt zerlegen: $27 = 2^4 + 2^3 + 2^1 + 2^0$. Daraus ergeben sich die Vorkommabits zu $0^{11}11011$. Zur Berechnung der Nachkommastellen verwenden wir wieder den oben beschriebenen Trick:

Stelle	Berechnung
0	$0{,}875 \cdot 2 = \mathbf{1}{,}75$
1	$0{,}750 \cdot 2 = \mathbf{1}{,}50$
2	$0{,}500 \cdot 2 = \mathbf{1}{,}00$
3	$0{,}000 \cdot 2 = \mathbf{0}{,}00$
⋮	⋮

Die Darstellung von 27,875 als Festpunktzahl lautet also:

$$0000\ 0000\ 0001\ 1011\ ,\ 1110\ 0000\ 0000\ 0000$$

b) Stellen Sie die beiden Zahlen aus Teilaufgabe a) möglichst genau als Gleitpunktzahlen gemäß IEEE-754 mit einfacher Genauigkeit dar.

Lösung:

- Darstellung von -12,43 als Gleitpunktzahl.

  Wir nutzen die Ergebnisse aus Teilaufgabe a). Da der größte Exponent 2^3 ist, kann man Zahlen bis 2^{-20} darstellen. Man muss also noch weitere Nachkommabits berechnen:

Stelle	Berechnung
15	$0{,}24 \cdot 2 = \mathbf{0}{,}48$
16	$0{,}48 \cdot 2 = \mathbf{0}{,}96$
17	$0{,}96 \cdot 2 = \mathbf{1}{,}92$
18	$0{,}92 \cdot 2 = \mathbf{1}{,}84$
19	$0{,}84 \cdot 2 = \mathbf{1}{,}68$
20	$0{,}68 \cdot 2 = \mathbf{1}{,}36$

  Demnach ergibt sich:

$$\begin{aligned}-12{,}43 =&(-1)^1 \cdot (2^3 + 2^2 + 2^{-2} + 2^{-3} + 2^{-5} \\ &+ 2^{-6} + 2^{-7} + 2^{-12} + 2^{-14} + 2^{-18} + 2^{-19} + 2^{-20} + 2^{-21}) \\ =&(-1)^1 \cdot 2^{130-127} \cdot (1 + 2^{-1} + 2^{-5} + 2^{-6} + 2^{-8} + 2^{-9} \\ &+ 2^{-10} + 2^{-15} + 2^{-17} + 2^{-21} + 2^{-22} + 2^{-23} + 2^{-24})\end{aligned}$$

  Es ist also $V = 1$, $C = 130_{10} = 10000010_2$ und $m' = 10001101110000101001000$ (da 2^{-24} ebenfalls ein Summand ist, runden wir auf). Demzufolge lautet die IEEE-754-Darstellung der Zahl:

$$1\ 10000010\ 10001101110000101001000$$

- Darstellung von 27,875 als Gleitpunktzahl.

$$\begin{aligned}27{,}875 =&(-1)^0 \cdot (2^4 + 2^3 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3}) \\ =&(-1)^0 \cdot 2^{131-127} \cdot (1 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7})\end{aligned}$$

Also ist $V = 0$, $C = 131_{10} = 10000011_2$ und $m' = 10111110000000000000000$. Somit lautet die IEEE-754-Darstellung der Zahl:

0 10000011 10111110000000000000000

c) Addieren Sie die beiden Zahlen aus den vorherigen Teilaufgaben jeweils in Festpunkt- bzw. Gleitpunkt-Darstellung.

Lösung:

- Addition in Festpunkt-Darstellung.

		1111	1111	1111	0011	,	1001	0001	1110	1100
+		0000	0000	0001	1011	,	1110	0000	0000	0000
	1	0000	0000	0000	1111	,	0111	0001	1110	1100

Unter Vernachlässigung der durch Übertrag entstandenen 1 erhalten wir die Festpunkt-Darstellung der Summe:

0000 0000 0000 1111 , 0111 0001 1110 1100

- Addition in Gleitpunkt-Darstellung.

Zur Addition in Gleitpunkt-Darstellung muss zunächst die Differenz der Exponenten betrachtet werden. Da der Exponent der zweiten Zahl um 1 größer ist als der Exponent der ersten Zahl, muss die Mantisse der zweiten Zahl um eins nach links verschoben werden. Zudem ist die erste Zahl negativ, weshalb wir sie von der zweiten Zahl binär subtrahieren. Vor beiden Mantissen muss die implizite 1 ergänzt werden. Die binäre Subtraktion der Mantissen ergibt sich zu:

	1	1011	1110	0000	0000	0000	0000
−		1100	0110	1110	0001	0100	1000
	0	1111	0111	0001	1110	1011	1000

Für die Charakteristik der neuen Zahl wird die größere der Charakteristiken der beiden Ausgangszahlen minus der Anzahl der führenden Nullen im Ergebnis der Subtraktion der Mantissen genommen, also $C_{add} = (131-1)_{10} = 130_{10} = 10000010_2$. Das Vorzeichen ist positiv, wie man durch einen Vergleich der Zahlen für die Mantissen-Subtraktion erkennen kann, also gilt $V_{add} = 0$. Die neue Mantisse ergibt sich, indem man aus dem Ergebnis der obigen Berechnung die führende Null und die dahinterstehende implizite Eins weglässt. Die IEEE-754 Darstellung der neuen Zahl lautet also:

0 10000010 11101110001111010111000

Aufgabe 50 ★

ZAH-AG Zahlendarstellung: BCD, Vorzeichen-Betrag, Eins-, Zweikomplement u. a.

Gegeben sei der folgende 32-Bit-Binärstring:

$$x = 1001\ 1001\ 0100\ 0000\ 0000\ 0000\ 0000\ 1000$$

Welcher Zahlenwert wird durch den Binärstring kodiert, wenn dieser interpretiert wird als

a) BCD-Zahl

Lösung:

99400008

b) Zahl zur Basis 2

Lösung:

$2^{31} + 2^{28} + 2^{27} + 2^{24} + 2^{22} + 2^3 = 2571108360$

c) Dualzahl in Vorzeichen-Betrag-Darstellung

Lösung:

$-(2^{28} + 2^{27} + 2^{24} + 2^{22} + 2^3) = -423624712$

d) Dualzahl in Einskomplement-Darstellung

Lösung:

$-(2^{31} - 1) + 2^{28} + 2^{27} + 2^{24} + 2^{22} + 2^3 = -1723858935$

e) Dualzahl in Zweikomplement-Darstellung

Lösung:

$-2^{31} + 2^{28} + 2^{27} + 2^{24} + 2^{22} + 2^3 = -1723858936$

f) Gleitpunktzahl in IEEE-754-Darstellung mit einfacher Genauigkeit

Lösung:

$$\underbrace{1}_{v}\ \underbrace{00110010}_{c}\ \underbrace{10000000000000000001000}_{m'}$$

$$\begin{aligned} V &= 1 \\ E &= 00110010_2 - 127 = 2^5 + 2^4 + 2^1 - 127 = 50 - 127 = -77 \\ M &= 100\ 0000\ 0000\ 0000\ 0000\ 1000_2 + 1 = 2^{-1} + 2^{-20} + 1 \end{aligned}$$

$$\begin{aligned} G(x) &= (-1)^V \cdot 2^E \cdot M = -2^{-77} \cdot (1 + 2^{-1} + 2^{-20}) \\ &\approx -9{,}926174 \cdot 10^{-24} \end{aligned}$$

Aufgabe 51 ★⋆

ZAH-AF **Zahlendarstellung: IEEE-754, BCD, Einskomplement**

a) Stellen Sie die Dezimalzahl −50,25 als Gleitpunktzahl mit einfacher Genauigkeit gemäß IEEE-754 dar.

Lösung:

- Zerlegung der Zahl in Vorzeichen und Zweierpotenzen:

$$-50{,}25 = (-1)^1 \cdot (32 + 16 + 2 + 1/4) = (-1)^1 \cdot (2^5 + 2^4 + 2^1 + 2^{-2})$$

- Ausklammern der größten Zweierpotenz; da für die Mantisse M gelten soll

$$1 \leq M < 2$$

(außer bei denormalisierten Zahlen mit $c = 0$), benötigen wir einen Wert zwischen 1 und 2 in der Klammer.

$$-50{,}25 = (-1)^1 \cdot 2^5(1 + 2^{-1} + 2^{-4} + 2^{-7})$$

Es gilt also für Vorzeichen V, Exponent E und Mantisse M:

$$V = 1, E = 5, M = 1 + 2^{-1} + 2^{-4} + 2^{-7}$$

- Da 8 Bits für die Charakteristik C zur Verfügung stehen, berechnet sie sich wie folgt:

$$\begin{aligned} q &= 2^{8-1} - 1 = 127 \\ C &= E + q = 132 \end{aligned}$$

Die Mantisse ohne implizite Eins m' ergibt sich zu $2^{-1} + 2^{-4} + 2^{-7}$.

- IEEE-754-Darstellung:

$$\begin{aligned} v &= 1, \\ m' &= 10010010000000000000000, \\ c &= 10000100 \end{aligned}$$

Daher:

$$\begin{aligned} -50{,}25_{10} &= (-1)^1 \cdot 2^{132-127}(1 + (2^{-1} + 2^{-4} + 2^{-7})) \\ &= 1\,10000100\,10010010000000000000000_{GPZ} \end{aligned}$$

b) Stellen Sie die Dezimalzahl 983400 als BCD-Zahl mit 32 Bits dar.

Lösung:

Die BCD-Kodierung ist eine Blockkodierung, bei der jede Ziffer dual mithilfe von 4 Bits dargestellt wird, d. h. insgesamt können mit 32 Bits Zahlen mit bis zu 8 Ziffern dargestellt werden. Zwei Stellen sind also unbesetzt, deshalb werden die linkesten 8 Bits auf Null gesetzt.

$$\begin{aligned} 9_{10} &= 1001_2 \\ 8_{10} &= 1000_2 \\ 3_{10} &= 0011_2 \\ 4_{10} &= 0100_2 \\ 0_{10} &= 0000_2 \\ 983400_{10} &= 0000\,0000\,1001\,1000\,0011\,0100\,0000\,0000_{BCD} \end{aligned}$$

c) Stellen Sie die Dezimalzahl -1236 als Einskomplement mit 16 Bits dar.

Lösung:

a) Der Betrag wird in Zweierpotenzen aufgespalten und dual dargestellt.

$$1236 = 1024 + 128 + 64 + 16 + 4 = 2^{10} + 2^7 + 2^6 + 2^4 + 2^2 = 0000010011010100_2$$

b) Nun werden zur Darstellung der Negation alle Bits gekippt, sodass sich folgende Lösung ergibt:

$$-1236_{10} = 1111101100101011_{1K,16}$$

Aufgabe 52 ⋆

ZAH-AC **Zahlendarstellung: Einskomplement, Zweikomplement, IEEE-754**

a) Geben Sie die Dezimalzahl -21 als Zahl in der Einskomplement-Darstellung mit 8 Bits an.

Lösung:

11101010

b) Geben Sie die Dezimalzahl -21 als Zahl in der Zweikomplement-Darstellung mit 8 Bits an.

Lösung:

11101011

c) Geben Sie den Dezimalwert $G(z)$ des folgenden Bitstrings z in IEEE-754-Darstellung mit einfacher Genauigkeit an.

$$z = 0\,10000011\,01010101000000000000000$$

Lösung:

Der Dezimalwert ergibt sich folgendermaßen:

$$\begin{aligned} G(z) &= (-1)^v \cdot 2^{c-q} \cdot (1 + m') \\ &= (-1)^0 (2^{131-127})(1 + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8}) \\ &= 21{,}3125 \end{aligned}$$

d) Welche Bedeutung hat bei der IEEE-754-Darstellung das implizite Bit der Mantisse und warum wird es nicht explizit dargestellt?

Lösung:

Das implizite Bit stellt die Ziffer vor dem Komma der normierten Gleitpunktzahl dar. In der Dualdarstellung steht vor dem Komma einer normierten Gleitpunktzahl immer eine 1 (allgemein steht dort in jeder Darstellung zur Basis b eine Ziffer zwischen 1 und $b-1$; für die Dualdarstellung bleibt also nur eine Ziffer übrig). Da die 1 immer vorhanden ist, muss man sie nicht explizit speichern und kann den Speicherplatz einsparen. Eine Ausnahme stellen die denormalisierten Zahlen dar, bei denen auf die Normierung verzichtet wird, um eine noch größere Genauigkeit im Zahlenbereich nahe der Null zu erreichen.

Aufgabe 53 ★⋆

VER-AC **Kodierung und Zahlendarstellung**

Geben Sie an, ob folgende Aussagen wahr oder falsch sind.

Lösung:

	Wahr	Falsch
Die BCD-Kodierung benötigt weniger Bits zur Speicherung einer Zahl als eine auf den Ziffern dieser Zahl basierende Huffman-Kodierung.	☐	☒

Erklärung: Die Huffman-Kodierung ist stets minimal, also kann die BCD-Kodierung nicht weniger Bits benötigen.

Zu einer gegeben n-Bit-Kodierung (also Zuordnung von Zeichen zu Bitfolgen) lässt sich ohne weitere Angaben eine äquivalente Huffman-Kodierung angeben.	☐	☒

Erklärung: Zusätzlich wird noch eine Häufigkeitsverteilung benötigt.

Beschränkt man die IEEE-754-Darstellung auf die Regel, dass Zahlen als $(-1)^v \cdot (1 + m')^{c-q}$ darzustellen sind (mit der bekannten Bedeutung der Variablen), so kann man zwar unendlich nicht mehr darstellen, die Null aber schon.	☐	☒

Erklärung: Weder unendlich noch die Null können dargestellt werden.

Die Subtraktion einer Zahl b von einer Zahl a erfolgt in der Zweikomplement-Darstellung durch die Addition des Komplements von b zu a und der anschließenden Addition von 1.	☐	☒

Erklärung: Die Subtraktion erfolgt einfach durch eine bitweise Addition von a und $-b$.

Die Kodierung der Telefonnummern beim Impulswahlverfahren erfüllt die Fano-Bedingung.	☒	☐
Die n-stellige Paritätsfunktion gibt an, ob in einem n-Bit-Wort $w \in \{0,1\}^n$ die Anzahl der Einsen derart ist, dass sich aus den Einsen Paare bilden lassen, ohne dass nach der Aufteilung eine Eins alleine übrigbleibt.	☒	☐

6 Rechnerarchitektur, Speicherorganisation und Internettechnologie

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=363

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 54 ★⋆

REK-AA **Von-Neumann-Rechner**

a) Nennen Sie die typischen Bestandteile heutiger Von-Neumann-Rechner.

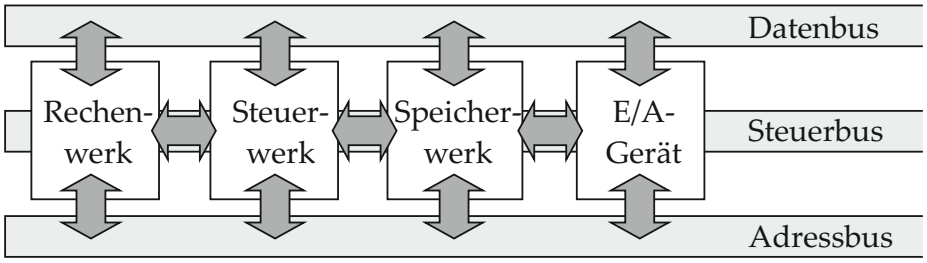
Lösung:

Die typischen Funktionseinheiten sind: Rechenwerk, Speicherwerk, Steuerwerk, Ein- und Ausgabewerk – über interne Datenwege (Busse) verbunden.

b) Wozu dienen im Allgemeinen die verschiedenen Busse? Skizzieren Sie die Busstruktur.

Lösung:

Die Busse dienen zur Übertragung von Daten, Befehlen, Kontroll- und Statusinformationen zwischen den Funktionseinheiten und stellen somit die Voraussetzung für deren Zusammenwirken dar. Skizze der Busstruktur:



c) Was versteht man unter dem “physikalischen Von-Neumann-Engpass”?

Lösung:

Der physikalische Von-Neumann-Engpass bezeichnet den Sachverhalt, dass das Verbindungssystem (Busse) zum Engpass zwischen CPU und Arbeitsspeicher wird. Der Transport einer Vielzahl von Daten zwischen CPU und Arbeitsspeicher dauert wesentlich länger als die Ausführung der Befehle durch die CPU.

d) Was verbirgt sich hinter dem Begriff des “intellektuellen Von-Neumann-Engpasses”?

Lösung:

Der intellektuelle Von-Neumann-Engpass bezieht sich auf die strenge Sequentialisierung aller Berechnungen. Die Programmierung muss demnach den sequentiellen Ablauf planen, was sich auch auf höhere Programmiersprachen auswirkt.

e) Nennen Sie 3 Konzepte zur Vermeidung des Von-Neumann-Engpasses.

Lösung:

Konzepte zur Vermeidung des Von-Neumann-Engpasses:

- Datentypen auf Hardware-Ebene, sogenannte 'Datenstruktur-Rechner';
- Assoziativer (inhaltsorientierter) Zugriff auf Daten (eher physikalisch);
- Parallelverarbeitung (beide).

f) Welches Konzept hilft bei der Lösung des bei Von-Neumann-Rechnern üblichen Problems der "semantischen Kluft" zwischen den Daten von Anwenderprogrammen und deren Repräsentation im Speicher? Welches Problem entsteht hingegen durch einen assoziativen Zugriff auf Daten?

Lösung:

Lösung des Problems der "semantischen Kluft": Unterstützung von Datenstrukturen durch die Hardware (beispielsweise Vektorrechner). Problem des assoziativen Zugriffs auf Daten: Komplexe Hardware (Vergleichslogik) ist erforderlich.

Aufgabe 55 ⋆

REK-AO **Rechnerarchitektur**

a) Nennen und charakterisieren Sie kurz die fünf Ebenen der Parallelverarbeitung. Welche dieser Ebenen zählt man zur feinkörnigen Parallelisierung und welche zur grobkörnigen?

Lösung:

Die folgenden Ebenen der Parallelverarbeitung werden unterschieden (von feinkörnig zu grobkörnig sortiert):

- **Ebene der Befehlsphasen** (Fetch, Decode, Execute, ...),
- **Ebene der Elementaroperationen** ($+$, $-$, $\cdot$, ...),
- **Anweisungs-Ebene** (Anweisungen von Programmiersprachen),

- **Prozess-Ebene** (Prozess = Zusammenfassung einer Anweisungsfolge zu einer funktionellen Einheit),
- **Job-Ebene** (Job = in sich abgeschlossenes, i. A. aus mehreren Prozessen bestehendes Benutzerprogramm).

Man spricht von **feinkörniger** Parallelisierung, wenn es sich um die Ebenen der Befehlsphasen, der Elementaroperationen oder der Anweisungen handelt. Zur **grobkörnigen** Parallelisierung zählt man die Prozess- und die Job-Ebene.

b) Erläutern Sie das Buszuteilungskonzept der parallelen Abfrage. Gehört es zu den zentralen oder dezentralen Steuerungen?

Lösung:

Jeder Sender hat eine eindeutige Priorität, die ihm selbst bekannt ist. Alle Sendewünsche sind über eine Meldeleitung für alle Sender sichtbar. Der Sender, der senden möchte und die höchste Priorität hat, darf senden. Jeder Sender weiß, ob er selbst gerade die höchste Priorität hat oder nicht, und sendet dementsprechend oder nicht. Die Parallele Abfrage ist eine **dezentrale Steuerung**.

c) Nennen Sie jeweils

- ein Eingabegerät, das kein Ausgabegerät ist,
- ein Ausgabegerät, das kein Eingabegerät ist und
- ein Gerät, das sowohl ein Ein- als auch ein Ausgabegerät ist.

Lösung:

Die folgenden Listen von Geräte stellen selbstverständlich nur Beispiele aus einer riesigen Auswahl an Ein- und Ausgabegeräten dar, die heute verfügbar sind:

- Reine Eingabegeräte: Maus, Tastatur;
- Reine Ausgabegeräte: Bildschirm, Drucker, Lautsprecher;
- Ein- und Ausgabegeräte: Festplatte, CD, DVD, USB-Stick.

Aufgabe 56 ★⋆

REK-AL **Rechnerarchitektur**

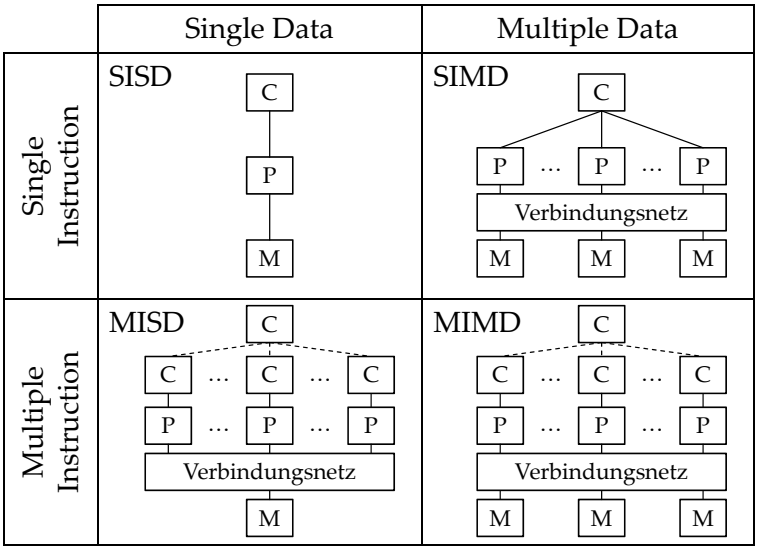
Rechnerarchitekturen können nach Flynn [1] bezüglich der

- Anzahl ihrer Steuerwerke (Befehlsströme) und
- Anzahl ihrer Speicherbereiche (Datenströme)

klassifiziert werden. Es ergeben sich vier grundsätzlich unterschiedliche Typen von Rechnerarchitekturen, siehe Abbildung. Skizzieren Sie für jeden dieser Typen die Anordnung der folgenden Blöcke:

- Steuerwerke (“Control” (C)),
- Rechenwerke (“Processor” (P)),
- Speicherwerke (“Memory” (M)).

Lösung:



Aufgabe 57 ⋆

REK-AQ **Multi-Threading**

a) Erklären Sie das Prinzip des simultanen Multi-Threadings (Hyperthreadings).

Lösung:

Ein Prozessor kann normalerweise in jedem Takt nur Befehle eines Threads verarbeiten, jedoch nutzt ein Thread nicht immer die gesamte Rechenleistung. Prozessoren mit simultanem Multi-Threading (Intel: Hyperthreading) haben mehrere Registersätze auf dem Prozessorchip und können dadurch gleichzeitig an mehreren Threads arbeiten. Das heißt die Rechenleistung wird nicht erhöht, aber besser ausgenutzt.

b) Welche zusätzlichen Kosten entstehen typischer Weise bei einem Prozessor, der simultanes Multi-Threading unterstützt?

Lösung:

Höhere Leistungsaufnahme; etwa 5% mehr Gatter.

c) Mit welcher Leistungssteigerung kann man beim Nutzen von simultanem Multi-Threading rechnen?

Lösung:

Bestenfalls ergibt sich bei zwei Threads die doppelte Leistung, wenn sich die Anforderungen der Threads gerade ergänzen. Schlimmstenfalls kann es allerdings zu einer Verlangsamung durch den Overhead kommen oder wenn beispielsweise parallele Threads jeweils den vollen Cache beanspruchen würden. Durchschnittlich erreicht man etwa 10 – 20% Leistungssteigerung.

Aufgabe 58 ★★⋆

REK-AB **Rechenwerk**

a) Das Rechenwerk ist ein Bestandteil der CPU in Computern. Additionswerke sind wiederum grundlegende Teile von Rechenwerken. Erläutern Sie, wie die mathematischen Operationen Subtraktion, Multiplikation und Potenzieren (auf Basis natürlicher Zahlen) auf die Addition zurückgeführt werden können.

In welchen Fällen würde man in einem realen Rechenwerk diese Rückführung auf die Addition nutzen? Begründen Sie.

Lösung:

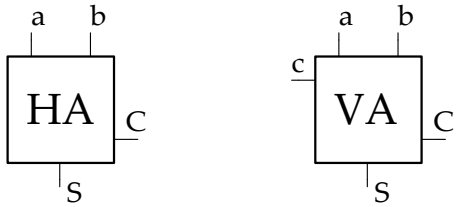
- Subtraktion: $a - b = a + (-b)$
- Multiplikation: $k \cdot n = \underbrace{k + k + \ldots + k}_{n\text{-mal}}$
- Potenzieren: $k^n = \underbrace{k + k + \ldots + k}_{k^{n-1}\text{-mal}}$

Multiplikation und Potenz würde man direkt berechnen, weil das für die meisten Zahlen sehr viel effizienter geht. Die Differenz wird für Zahlendarstellungen, bei denen die Negation leicht zu berechnen ist, tatsächlich auf die Addition zurückgeführt (siehe Eins- und Zweikomplement-Darstellung).

b) Im Rechenwerk wird die Addition durch Addierer (engl. “Adder”) ausgeführt. Erläutern Sie am Beispiel der Addition von zwei 32-Bit-Zahlen in Dualdarstellung die Unterschiede zwischen Halbaddierern und Volladdierern.

Lösung:

Sowohl Halbaddierer (HA) als auch Volladdierer (VA) berechnen ein Summenbit S und ein Übertragsbit C (englisch “carry”) für eine 1-Bit-Addition. Im Unterschied zu HA können VA einen eventuellen Übertrag aus einer vorangegangenen Addition zweier Bits mit beachten. Aus diesem Grund hat der HA zwei Eingänge für die zu addierenden Bits a und b, und der VA einen zusätzlichen Eingang für den Übertrag c der vorherigen Addition.



Eine Addition zweier 32-Bit-Zahlen benötigt daher einen HA sowie 31 VA. Erfolgt die Addition "von rechts nach links", so werden in einem ersten Schritt die niederwertigsten Bits der zwei 32-Bit-Zahlen im HA addiert. Da dabei ein Übertrag anfallen kann, werden für die 31 übrigen Bits VA zur Addition benötigt.

c) Entwickeln Sie die Schaltfunktion des Halbaddierers als Wahrheitstabelle und Booleschen Ausdruck.

Lösung:

Seien a und b die zu addierenden Bits, S ihre Summe und C der Übertrag der Addition der beiden Bits.

Schaltfunktion als Wahrheitstabelle:

a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

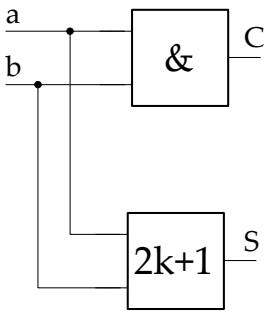
Schaltfunktion als Boolescher Ausdruck:

$$S = (a \cdot b') + (a' \cdot b) = a \oplus b \text{ (XOR-Funktion)}$$
$$C = a \cdot b$$

d) Entwickeln Sie für den Halbaddierer eine Schaltung auf Gatterebene auf Grundlage Ihrer Ergebnisse aus Aufgabenteil c). Verwenden Sie dabei nur die Gatter AND, OR, NOT, NAND, NOR und XOR.

Lösung:

Schaltung auf Gatterebene:



Aufgabe 59 ★⋆

REK-AM **Speicherwerk**

Die heute verfügbaren Realisierungen von Speicherwerken unterscheiden sich grundlegend in Art und Geschwindigkeit des Datenzugriffs sowie in der Art der Datenhaltung.

a) Beschreiben Sie Unterschiede und Gemeinsamkeiten der Datenspeichertypen RAM und ROM. Was bedeuten diese Abkürzungen?

Lösung:

RAM: Random Access Memory; ROM: Read Only Memory.

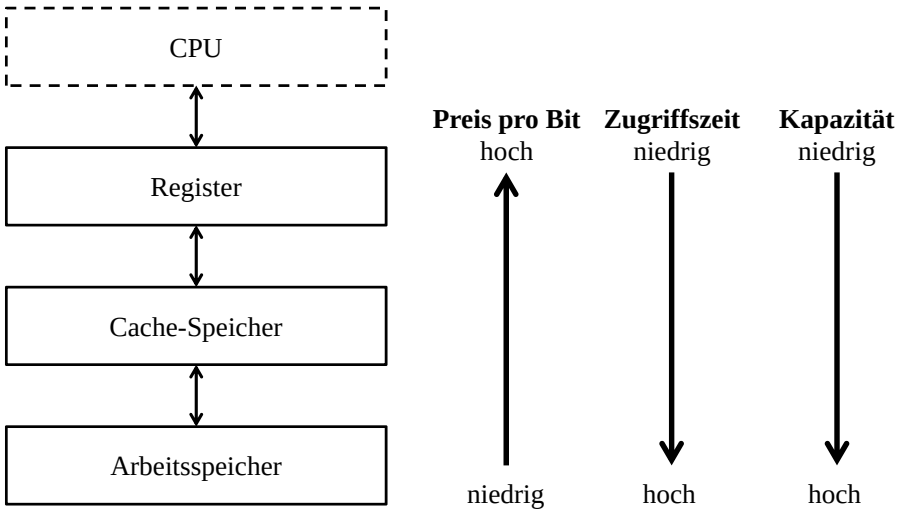
- Gemeinsamkeiten: Beide sind gewöhnlich Speicher mit wahlfreiem Zugriff – obwohl die explizite Nennung von "Random Access" bei RAM suggerieren könnte, dass das nur dort der Fall sei (beachte jedoch die Ausnahme der "CD-ROM", die einen blockadressierbaren Speicher darstellt).
- Unterschiede: Als RAM bezeichnet man Speicherbausteine, die gelesen und beschrieben werden können und die gewöhnlich flüchtig sind. ROM-Speicher können nur gelesen werden (manchmal auch mit größerem Aufwand beschrieben) und sie sind deshalb natürlich nicht flüchtig.

b) Sortieren Sie Register, Cache-Speicher und Arbeitsspeicher nach

- der Zugriffszeit für die CPU,
- dem Preis pro Bit und
- der daraus resultierenden üblichen Speicherkapazität in modernen Rechnern

auf einer relativen Skala von "hoch" bis "niedrig" ein.

Lösung:



Aufgabe 60 ★

REK-AE **Cache**

a) Welchen Zweck erfüllt ein Cache-Speicher? Nennen Sie in dem Zusammenhang den Vorteil beim Zugriff auf den Cache gegenüber dem Zugriff auf einen Hauptspeicher.

Lösung:

Der Cache ist ein schneller, aber meist kleiner Zwischenspeicher, der Daten (Adressen, Befehle, ...) speichert, auf die zuvor zugegriffen wurde und von denen erwartet wird, dass sie in naher Zukunft wieder benötigt werden. Er ermöglicht einen sehr viel schnelleren Zugriff auf Daten, die bereits einmal vorlagen (und noch nicht wieder entfernt wurden), als der Hauptspeicher, ist dafür aber deutlich teurer pro Bit.

b) Beschreiben Sie zwei Zugriffsarten auf den Cache und nennen Sie je einen Vor- und einen Nachteil.

Lösung:

- **Directly-mapped**: Der Cache enthalte k Blöcke, also k Speicherpositionen mit Tag- und Datenfeld, deren Adressen von 0 bis $k-1$ durchnummeriert seien (üblicherweise mit $k = 2^n$ für $n \in \mathbb{N}$). Der Datenblock von Adresse b aus dem Hauptspeicher gelangt in das Datenfeld an Position $b \bmod k$ im Cache. Im Tag-Feld wird dort $b \operatorname{div} k$ gespeichert.

  Vorteil: Die CPU muss jeden Block nur an einer Stelle im Cache suchen, ohne dass ein echter Assoziativspeicher benötigt wird (ein wahlfreier Zugriff auf jedes Element ist ausreichend). **Nachteile**: Bei ungünstigen "Zugriffsmustern" kann eine ungünstige Auslastung des Caches durch Kollisionen entstehen; die Daten werden, unabhängig von ihrer erwarteten Nutzung, nur solange im Cache behalten, bis zufällig eine Adresse denselben Block beansprucht.

- **Assoziativ**: Ein Block aus dem Hauptspeicher kann an eine beliebige Position im Cache gelangen, unabhängig von seiner Hauptspeicheradresse. Beispielsweise wird beim "Least-Recently-Used"-Prinzip immer der am längsten nicht benutzte Block im Cache durch den neuen Block aus dem Hauptspeicher überschrieben.

  Vorteile: Der Cache wird bestmöglich ausgenützt und ist (nach einer Anfangsphase) immer komplett gefüllt; die zu erwartende Anzahl der Cache-Fehler wird dadurch reduziert; die erwartete Nutzung der Daten wird in den Überschreibungsprozess einbezogen (Daten, die lange nicht genutzt wurden, haben eine geringere Wahrscheinlichkeit, bald benötigt zu werden, als Daten, die vor kurzem genutzt wurden). **Nachteil**: Ein echter Assoziativspeicher wird benötigt, da die CPU sonst alle Felder sequentiell nach einem Block durchsuchen müsste, was viel zu lange dauern würde; echte Assoziativspeicher können parallel alle Felder in einem Takt durchsuchen, kosten aber wegen der komplexen Schaltlogik mehr als gewöhnliche wahlfreie Speicher.

Aufgabe 61 ⋆

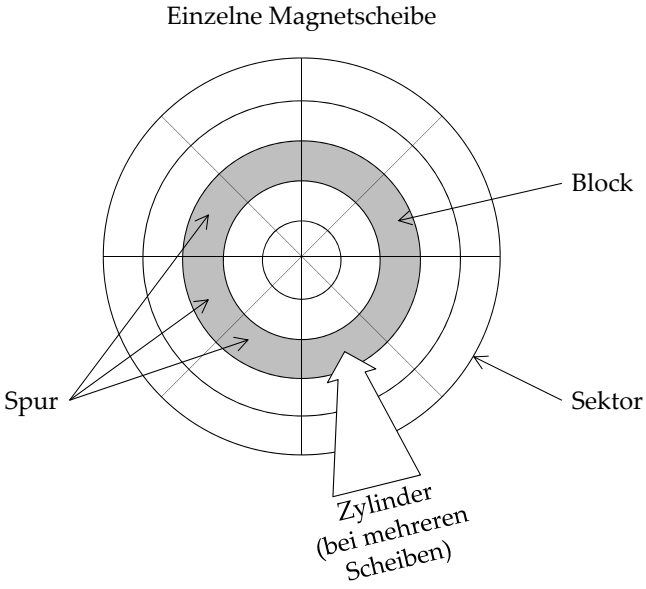
REK-AP **Hintergrundspeicher**

Beschreiben Sie den Aufbau einer Magnetfestplatte (Hintergrundspeicher) und erläutern Sie allgemein, aus welchen Anteilen sich die Zugriffszeiten auf Daten, die sich auf der Festplatte befinden, zusammensetzen.

Lösung:

Die Magnetfestplatte besteht aus mehreren Magnetscheiben, welche auf beiden Seiten beschichtet sind und daher auch beidseitig beschrieben werden können. Lese- und Schreibköpfe sorgen für den Zugriff auf die einzelnen Zylinder (übereinanderliegende Spuren bei mehreren Magnetscheiben). Der Zugriff auf die einzelnen Blöcke erfolgt mittels einer Adresse, die sich aus folgenden Daten zusammensetzt:

- Zylindernummer,
- Spurnummer,
- Sektorennummer.



Zugriffszeiten auf einen Block:

- Zeit für die mechanische Positionierung der Lese-/Schreibköpfe auf den gesuchten Zylinder (besonders zeitaufwändig);
- Auswahl der Spurnummer;
- Wartezeit, bis gesuchter Sektor vorbeikommt (Dauer, bis sich die Magnetplatte auf den richtigen Sektor gedreht hat).

Aufgabe 62 ⋆

REK-AU **Speicherorganisation**

a) Was ist der Arbeitsspeicher bzw. Hauptspeicher? Wie erfolgt der Zugriff auf ein Speicherelement?

Lösung:

Der Arbeitsspeicher setzt sich aus einer Folge gleichgroßer, durchnummerierter Speicherelemente, den kleinsten adressierbaren Einheiten, zusammen. Jedes Speicherelement hat eine eindeutige Adresse (beachte Unterscheidung virtuelle vs. reale Adresse). Über die Adresse kann unmittelbar (wahlfrei) auf ein Speicherelement zugegriffen werden (daher “Random Access Memory (RAM)”).

b) Über welche drei speziellen Register erfolgt die Kommunikation mit dem Arbeitsspeicher?

Lösung:

- **Address-Memory-Port (AM)**: Enthält die Hauptspeicheradresse des Speicherelements, in das geschrieben bzw. aus dem gelesen werden soll.
- **Write-Memory-Port (WM)**: Enthält den Inhalt, der in das durch AM adressierte Speicherelement geschrieben werden soll.
- **Read-Memory-Port (RM)**: Enthält den aktuell gelesenen Inhalt des durch AM adressierten Speicherelements.

Aufgabe 63 ★⋆

REK-AG **Speicherorganisation und Adressierungsarten**

a) Suchen Sie die falschen Aussagen heraus und korrigieren Sie diese.

1) Wahlfreier Zugriff bedeutet, dass jedes Speicherelement unabhängig von seiner Position mit dem gleichen zeitlichen Aufwand direkt erreichbar ist. Ein Beispiel dafür ist der Magnetbandspeicher.
2) Beim blockadressierbaren Zugriff wird der Speicher in Blöcke eingeteilt. Der Zugriff auf diese Blöcke ist sequentiell, ebenso der Zugriff innerhalb der Blöcke.
3) Beim Directly-Mapped-Cache mit maximal k Blöcken gelangt der Block mit Adresse b an Position $b \bmod k$ im Cache; dieser Wert wird auch im Tag-Feld an dieser Position gespeichert.
4) Ein KiB sind 1024 Byte.
5) Die Aufgaben des Steuerwerks sind das Holen und Entschlüsseln von Befehlen, sowie die Initiierung der Befehlsausführung.

Lösung:

1) Falsch: Die Definition stimmt, das Beispiel ist allerdings falsch, korrekte Beispiele sind ROM und RAM.
2) Falsch: Die Einteilung in Blöcke ist korrekt, der Zugriff auf Blöcke ist jedoch wahlfrei, innerhalb der Blöcke ist der Zugriff sequentiell.
3) Falsch: Gespeichert wird im Tag-Feld $b \operatorname{DIV} k$.
4) Korrekt.
5) Korrekt.

b) Eine CPU, die über einen 9-zeiligen Directly-Mapped-Cache verfügt, ruft nacheinander Daten von folgenden Adressen auf:

$$85, 77, 13, 8, 90, 27, 110, 46, 38, 69$$

Geben Sie an, welche Einträge sich am Ende der Aufrufe in den Tag-Feldern des Caches befinden.

Lösung:

Aufgerufene Adressen: 85, 77, 13, 8, 90, 27, 110, 46, 38, 69. Die folgenden Daten befinden sich zum Schluss im Cache (durchgestrichen sind Daten zu temporär gespeicherten Adressen, die bis zum Schluss wieder überschrieben wurden):

Cache-Adresse	Tag	Datenblock
0	~~90 div 9~~ 27 div 9	...
1	46 div 9	...
2	~~110 div 9~~ 38 div 9	...
3	–	...
4	~~85 div 9~~ 13 div 9	...
5	77 div 9	...
6	69 div 9	...
7	–	...
8	8 div 9	...

c) Warum sind 9 Zeilen für einen Directly-Mapped-Cache untypisch?

Lösung:

Siehe Aufgabe REK-AF, b), Seite 237.

Aufgabe 64 ★⋆

REK-AH **Speicherorganisation und Adressierungsarten**

Eine CPU, welche über einen 8-zeiligen Directly-Mapped-Cache verfügt, ruft nacheinander Daten aus dem Hauptspeicher von folgenden Adressen auf:

$$3, 15, 8, 7, 6, 3, 2, 21, 33, 64, 8, 31, 11, 14$$

a) Geben Sie für jede der Zeilen im Cache an, welcher Eintrag dort am Ende der Aufrufe im Tag-Feld steht.

Lösung:

Der Cache enthält 8 Speicherpositionen, und der Block mit der Adresse z gelangt an die Position $z \bmod 8$ im Cache, wobei im Tag-Feld der Wert $z \operatorname{div} 8$ abgelegt wird.

	Tag	Datenblock
0	8 div 8	...
1	33 div 8	...
2	2 div 8	...
3	11 div 8	...
4	(leer)	...
5	21 div 8	...
6	14 div 8	...
7	31 div 8	...

b) Welche Werte liest der Befehl LOAD k für $k \in \{0, \ldots, 7\}$ jeweils aus dem folgenden Hauptspeicher aus, wenn es sich um **unmittelbare**, **direkte**, **indirekte** oder **indizierte** Adressierung handelt? Das **Indexregister** enthalte den Wert 1 (vgl. zu Adressierungsarten Kapitel 7).

Hauptspeicheradresse	Datum
0	5
1	7
2	6
3	3
4	7
5	1
6	0
7	2

Lösung:

k	unmittelbar	direkt	indirekt	indiziert
0	0	5	1	7
1	1	7	2	6
2	2	6	0	3
3	3	3	3	7
4	4	7	2	1
5	5	1	7	0
6	6	0	5	2
7	7	2	6	–

Aufgabe 65 ★★⋆

REK-AF **Speicherorganisation und Adressierungsarten**

Eine CPU verfügt über einen Cache mit 5 Blöcken. Jeder Block ist nochmal in 3 Zeilen unterteilt. Ruft die CPU eine Adresse a auf, wird zuerst über das Directly-Mapped-Verfahren entschieden, in welchen Block die Adresse gelangt. Innerhalb des Blocks wird über das Assoziativ-Verfahren mit Least-Recently-Used Prinzip entschieden, in welche Zeile die Adresse gelangt.

a) Die CPU ruft nacheinander folgende Adressen auf:

$$67, 49, 156, 95, 12, 38, 128, 92, 106, 78, 117, 38, 67, 45,$$
$$128, 38, 60, 53, 3, 78, 63, 206, 175, 38, 184, 156, 211$$

Geben Sie für jede Zeile in jedem Block an, welcher Wert sich am Ende im Tag-Feld befindet (vgl. Abbildung).

Lösung:

Block 0	~~95 div 5 (4)~~ 175 div 5 (23)
	45 div 5 (14)
	60 div 5 (17)
Block 1	156 div 5 (3) (26)
	~~106 div 5 (9)~~ 211 div 5 (27)
	206 div 5 (22)
Block 2	~~67 div 5 (1)~~ 117 div 5 (11)
	~~12 div 5 (5)~~ 67 div 5 (13)
	92 div 5 (8)
Block 3	~~38 div 5 (6) (12) (16)~~ 78 div 5 (20)
	~~128 div 5 (7) (15)~~ ~~3 div 5 (19)~~ 38 div 5 (24)
	~~78 div 5 (10)~~ ~~53 div 5 (18)~~ 63 div 5 (21)
Block 4	49 div 5 (2)
	184 div 5 (25)

b) Warum sind 5 Zeilen für einen Directly-Mapped-Cache untypisch?

Lösung:

Da es in Dualdarstellung sehr einfach ist, $x \bmod k$ und $x \operatorname{div} k$ zu berechnen, wenn $k = 2^i$ für ein $i \in \mathbb{N}$, nimmt man als Cache-Größe üblicherweise Zweierpotenzen. Beispiel:

$$19 = \boxed{10}\boxed{011} \text{ (Dualdarstellung)}$$

mit $10_2 = 19 \operatorname{div} 8$ und $011_2 = 19 \bmod 8$, wobei wir von hinten drei Stellen abspalten, weil $8 = 2^3$. (Analog verhält es sich im Dezimalsystem mit Zehnerpotenzen.)

Aufgabe 66 ★⋆

REK-AS **Bussysteme**

Beantworten Sie die folgenden Fragen zur Funktionsweise von Bussystemen.

a) Was ist die Kernaufgabe von Bussen?

Lösung:

Busse sind Datensammelwege zur Übertragung von Daten, Adressen oder Kontrollbefehlen und Statusinformationen.

b) Was versteht man unter der Breite eines Busses?

Lösung:

Die Breite eines Busses, oder auch die Bandbreite, beschreibt die Anzahl der parallel übertragbaren Bits, bzw. die Anzahl der Leitungen.

c) Was versteht man unter einem Datenbus?

Lösung:

Ein Datenbus überträgt Daten zwischen Computerbestandteilen oder Computern (üblicherweise) bidirektional. Die Breite eines Datenbusses entspricht normalerweise der Anzahl der Bits eines Arbeitsspeicherelements, damit jeweils ein solches Element auf einmal übertragen werden kann.

d) Was versteht man unter einem Adressbus?

Lösung:

Ein Adressbus wird benutzt um von der CPU berechnete Speicheradressen zu übertragen. Die auf dem Adressbus übertragene Adresse gibt die Quelle, bzw. das Ziel der Übertragung des aktuellen Datums auf dem Datenbus an. Adressbus und Datenbus werden zusammen benutzt, um die CPU mit Informationen für die durchzuführenden Berechnungen zu versorgen.

e) Was versteht man unter einem Kontrollbus oder Steuerbus?

Lösung:

Ein Steuerbus übermittelt Steuersignale zwischen der CPU und anderen Funktionseinheiten.

Aufgabe 67 ★⋆

REK-AR **Bussysteme**

Nachfolgend sind Aussagen über Bussysteme aufgeführt. Entscheiden Sie zu jeder, ob sie wahr oder falsch ist, und begründen Sie Ihr Urteil bei allen falschen Aussagen.

Lösung:

	Wahr	Falsch
Busse lassen sich nach verschiedenen Kriterien klassifizieren: Nach der Breite eines Busses, nach seiner Funktion, nach der Betriebsart (synchron/asynchron) und nach Typ (Peripheriebus/Prozessorbus).	☒	☐
Die Breite eines Busses gibt an, wie viele Steuersignale zwischen CPU und anderen Funktionseinheiten ausgetauscht werden.	☐	☒

Erklärung: Die Breite eines Busses, auch Bandbreite genannt, gibt die Anzahl der parallel übertragbaren Bits an.

Mögliche Funktionen von Bussen können Datenübertragung, Pipelining oder Übertragung der von der CPU berechneten Speicheradressen sein.	☐	☒

Erklärung: Pipelining stellt keine Funktion von Bussen dar, stattdessen müsste zusätzlich die Funktion des Kontrollbusses aufgeführt sein: Übermittlung von Steuersignalen.

Die parallele Abfrage lässt den Sender mit höchster Priorität senden.	☒	☐
Sowohl bei dem CSMA-Verfahren als auch bei der zyklischen Buszuteilung ist es nicht die Aufgabe der Bussteuereinheit Teilnehmerwünsche abzufragen (polling).	☒	☐
Bei Daisy Chain koordiniert die Bussteuereinheit die Übertragung.	☐	☒

Erklärung: Jeder Sender muss für sich entscheiden, ob der Bus verfügbar ist.

Unidirektionale Datenübertragung weist eine erheblich geringere Übertragungsgeschwindigkeit auf und ist deswegen eine ineffiziente Kommunikationsmöglichkeit.	☐	☒

Erklärung: Die Unidirektionale Datenübertragung besitzt eine erheblich höhere Übertragungsgeschwindigkeit und stellt deswegen eine besonders effiziente Kommunikationsmöglichkeit dar.

Aufgabe 68 ⋆

REK-AC **Buszuteilung**

a) Nennen Sie ein Buszuteilungsverfahren mit höherem und eines mit niedrigerem Verwaltungsaufwand (relativ zueinander) und begründen Sie Ihre Wahl.

Lösung:

- Hoher Aufwand: zentrale Verwaltungsstelle (die beispielsweise Polling betreibt); eigene Meldeleitung für jeden Teilnehmer.
- Niedriger Aufwand: Token-Ring, CSMA

Die letztgenannten Verfahren benötigen keine Zentralstelle und koordinieren sich über die Datenleitung, sie benötigen also keine Meldeleitung (andererseits setzen Token-Ring und CSMA dafür in den Teilnehmern mehr "Intelligenz" voraus).

b) Welches der Verfahren

- zyklische Buszuteilung unabhängig vom Übertragungswunsch,
- zirkulierendes Senderecht (Token-Ring) oder
- CSMA

nutzt den Bus am effizientesten?

Lösung:

Zirkulierendes Senderecht. Dieses Verfahren ist effizienter als die zyklische Buszuteilung, da kein Leerlauf autritt, wenn ein Teilnehmer nicht senden möchte. Außerdem ist es effizienter als CSMA, da es bei CSMA zu Kollisionen kommen kann, die zu Übertragungswiederholungen führen. Trotzdem hat sich Ethernet (und nicht Token-Ring) als Standardtechnologie durchgesetzt, nicht zuletzt weil die Verwaltung des Senderechts aufwändig ist.

Aufgabe 69 ★⋆

REK-AT **Pipelining**

In der Universität zu Musterstadt gibt es in der kleinen und stets überfüllten Mensa mit nur einer Essensausgabe fünf Stationen. Pro Station kann immer nur ein Student stehen und andere Kommilitonen können nicht überholt werden. Sobald eine Station frei wird, rückt der Student nach, der in der jeweils dahinterliegenden Station steht. Jeder Student beginnt bei der ersten Station und durchläuft alle Stationen. Eine **Salatbar** bildet die erste, eine **Suppenausgabe** die zweite Station. Es folgen eine Station zur Ausgabe der **Hauptspeise**, eine Auswahl verschiedener **Desserts** sowie als letztes die **Kasse**. Die durchschnittliche Verweildauer an der Salatbar beträgt **10 Sekunden**, an der Suppenausgabe **3 Sekunden**, an der Station zur Ausgabe der Hauptspeise **22 Sekunden**, zur Auswahl des Desserts **6 Sekunden** und an der Kasse **12 Sekunden**. Zur Vereinfachung der Aufgabe sollen die einzelnen Verweildauern als konstant angesehen werden.

a) Angenommen, die Mensa würde das Essen statt an fünf nur an einer Station ausgeben, und erst wenn ein Student komplett fertig wäre, könnte der nächste beginnen sich Essen zu nehmen. Wie lange würde es bei den gegebenen Verweildauern dauern bis ein Student sein Essen erhalten und bezahlt hat? Wie viele Studenten könnten pro Stunde maximal bedient werden?

Lösung:

Zeit für Essensausgabe pro Student: $10s + 3s + 22s + 6s + 12s = 53s$. Nach 53 Sekunden hat ein Student sein Essen erhalten und bezahlt und macht Platz für den nächsten Studenten. Maximale Anzahl an Studenten pro Stunde: $^{3600s}/_{53s} \approx 67$.

b) In welchem Abstand verlassen die Studenten im Betrieb mit fünf Stationen die Essensausgabe? Wie viele Studenten können pro Stunde maximal bedient werden?

Lösung:

Die höchste Verweildauer an den einzelnen Stationen gibt an, in welchem Abstand die Studenten die Essensausgabe verlassen. Der Zeitabstand beträgt somit 22 Sekunden, was der Verweildauer an der Station zur Ausgabe der Hauptspeise entspricht. Maximale Anzahl an Studenten pro Stunde: $^{3600s}/_{22s} \approx 163$.

c) Wie viel Zeit muss ein Student im Betrieb mit fünf Stationen von Beginn der Auswahl des Salats bis zum Verlassen der Essensausgabe einplanen?

Lösung:

Im Gegensatz zum Pipeline-Rechner werden die Studenten nicht von einem globalen Taktgeber gesteuert, d. h. nach Erhalten der Hauptspeise (Station mit höchster Verweildauer) kann ohne Wartezeit ein Dessert ausgewählt und bezahlt werden. Bevor man die Station mit der Hauptspeise (22 Sekunden Wartezeit) passiert hat, muss man an jeder Station 22 Sekunden warten, da es sich bei der Hauptspeise staut. Einzuplanende Zeit: $3 \cdot 22s + 6s + 12s = 84s$.

d) Aufgrund zahlreicher Beschwerden von Studenten, die Essensausgabe würde zu lange dauern und zu sehr langen Warteschlangen führen, entschließt sich die Mensaleitung, das Personal an der Station zur Ausgabe der Hauptspeise zu verdoppeln. Durch diese Personalaufstockung kann die Verweildauer an der Station zur Ausgabe der Hauptspeise halbiert werden.

- In welchem Abstand verlassen die Studenten nun die Essensausgabe? Wie viele Studenten können pro Stunde maximal bedient werden?
- Wie viel Zeit kann ein Student prozentual von Beginn der Auswahl des Salats bis zum Verlassen der Essensausgabe durch die Personalaufstockung einsparen?

Lösung:

- Die höchste Verweildauer an einer einzelnen Station liegt nun mit 12 Sekunden bei der Kasse, der Zeitabstand der Studenten beim Verlassen der Essensausgabe beträgt somit 12 Sekunden. Maximale Anzahl an Studenten pro Stunde: $^{3600s}/_{12s} = 300$.
- Da die Kasse als Station mit der höchsten Verweildauer das letzte Glied in der Kette der Stationen bildet und die Studenten an den einzelnen Stationen jeweils nur weiter rücken können, wenn der Kommilitone vor ihnen zur nächsten Station rückt, ist die Situation mit einem Pipeline-Rechner mit globalem Taktgeber vergleichbar. Taktperiode Pipeline-Architektur (globaler Taktgeber): $5 \cdot 12s = 60s$. Zeitliche Einsparung in Prozent: $^{84s-60s}/_{84s} \approx 28{,}6\%$.

Aufgabe 70 ⋆

REK-AD **Pipelining**

Tragen Sie die Ausführungsabfolge der Verarbeitung dreier Befehle eines Pipeline-Rechners in eine Tabelle ein (siehe Abbildung). Gehen Sie davon aus, dass hierbei drei Befehle ausgeführt werden müssen, die sich jeweils in die Bearbeitungsphasen $F \rightarrow D \rightarrow O \rightarrow E$ (Fetch → Decode → Get Operand → Execute) aufteilen. Geben Sie auch die Anzahl der Takte an, die mit Pipelining benötigt werden und die Ersparnis gegenüber einer Ausführung ohne Pipelining.

Lösung:

Takt	F	D	O	E
1	F_1	–	–	–
2	F_2	D_1	–	–
3	F_3	D_2	O_1	–
4	–	D_3	O_2	E_1
5	–	–	O_3	E_2
6	–	–	–	E_3

Anzahl der Takte mit Pipelining: 6; Anzahl der Takte ohne Pipelining: $3 \cdot 4 = 12$; Ersparnis: 6 Takte.

Aufgabe 71 ★

REK-AN **Pipelining**

Veranschaulichen Sie anhand eines selbst gewählten Beispiels, dass Folgendes gilt: Sei n die Anzahl der Teilaufgaben einer Befehlsverarbeitung und m die Anzahl der zu verarbeitenden Befehle, dann kann durch Befehlsphasen-Pipelining die Taktzahl T, die zur vollständigen Verarbeitung der Befehle nötig ist, von

$$T = n \cdot m$$

auf

$$T = n + m - 1$$

reduziert werden.

Lösung:

Betrachte zum Beispiel die Verarbeitung von drei Befehlen, die jeweils aus 3 Teilaufgaben bestehen, beispielsweise “Befehl holen”, “Dekodieren” und “Ausführen”. Man benötigt ohne Pipelining $T = 3 \cdot 3 = 9$ Takte zur Verarbeitung, während mit Befehlsphasen-Pipelining die Anzahl der Takte auf $T = 3 + 3 - 1 = 5$ reduziert werden kann, siehe Abbildungen.

Ohne Pipelining:

Takt	Befehl holen	Dekodieren	Ausführen
1	H1	–	–
2	–	D1	–
3	–	–	A1
4	H2	–	–
5	–	D2	–
6	–	–	A2
7	H3	–	–
8	–	D3	–
9	–	–	A3

Mit Pipelining:

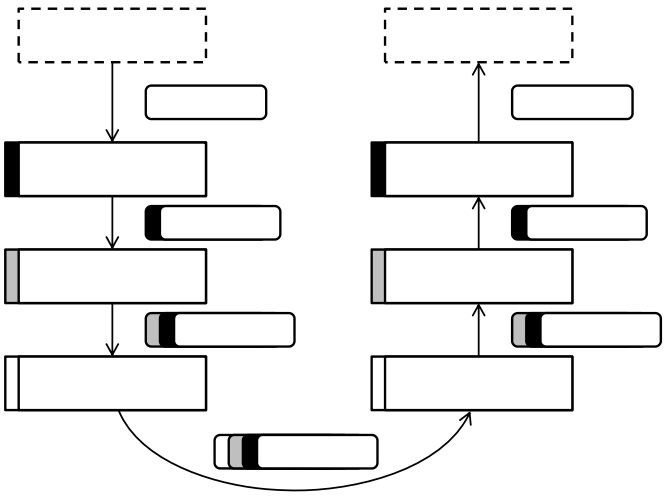
Takt	Befehl holen	Dekodieren	Ausführen
1	H1	–	–
2	H2	D1	–
3	H3	D2	A1
4	–	D3	A2
5	–	–	A3

Durch eine Verallgemeinerung dieser Einsicht sieht man, dass sich die Anzahl der Takte von $3 \cdot m$ auf $3 + m - 1$ bei m zu verarbeitenden Befehlen reduziert, bzw. bei n Teilaufgaben pro Befehl von $n \cdot m$ auf $n + m - 1$ Takte.

Aufgabe 72 ★

REK-AI **Schichtenmodel, TCP, IP, UDP**

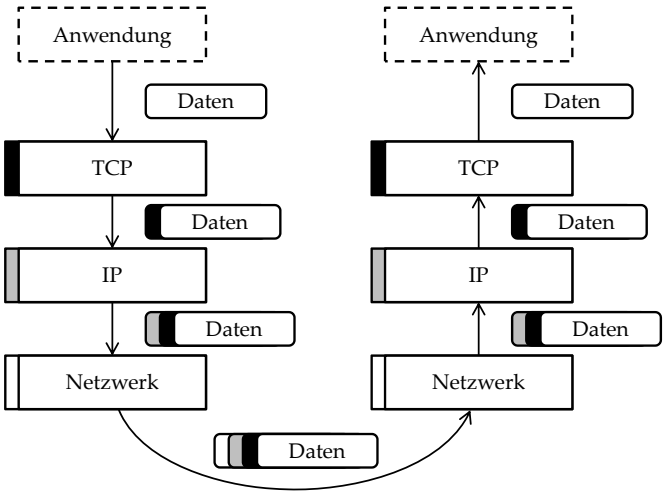
a) Gegeben sei ein vereinfachtes unausgefülltes Schaubild zum Schichtenmodell der Kommunikationsprotokolle in Rechnernetzen:



Tragen Sie die in der Tabelle angegebenen Begriffe in die Kästchen des Schaubilds ein.

Begriff	Beschreibung
Daten	Informationen, welche von einer Anwendung zu einer anderen übertragen werden sollen.
Netzwerk	Die physikalische Netzwerkschicht.
IP	Das **Internet Protocol (IP)** ist ein Netzwerkprotokoll der Vermittlungsschicht.
Anwendung	Anwendungsschicht, welche die Endanwendung und die Middleware enthält.
TCP	Das **Transmission Control Protocol (TCP)** ist ein Protokoll der Transportschicht.

Lösung:



b) Ordnen Sie folgenden Aussagen die Begriffe TCP, IP und/oder UDP zu.

- Dieses Protokoll stellt eine logische und verlässliche Verbindung zwischen Sender und Empfänger her.
- Dieses Protokoll sorgt für den unzuverlässigen, verbindungslosen Versand von Datenpaketen.
- Dieses Protokoll wird beispielsweise für Multi-Media-Übertragungen eingesetzt.
- Diese beiden Protokolle (der angegebenen drei) befinden sich auf derselben Ebene des OSI-Schichtenmodells.

Lösung:

- **TCP** stellt eine logische, verlässliche Verbindung zwischen Sender und Empfänger her.

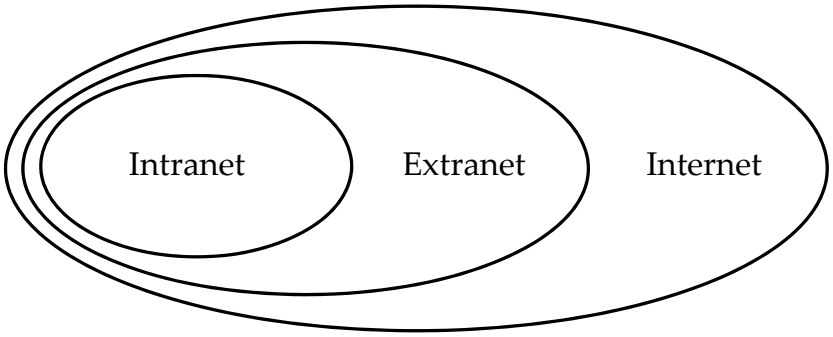
- **IP** sorgt für den unzuverlässigen, verbindungslosen Versand von Datenpaketen.
- **UDP** wird für Datenströme eingesetzt, bei denen vereinzelt fehlerhaft übertragene Pakete in Kauf genommen werden können, beispielsweise bei Multi-Media-Übertragungen.
- **TCP** und **UDP** befinden sich beide in der Transportschicht.

Aufgabe 73 ⋆

REK-AJ **Netze**

Beschreiben Sie die verschiedenen Typen von Netzen (Internet, Intranet und Extranet) und grenzen Sie die Begriffe voneinander ab. Nutzen Sie eine Skizze.

Lösung:



Die Netze bilden eine hierarchische Struktur, wie in der Abbildung angedeutet.

- **Intranet**: Privates/firmeninternes Netz, von außen kein Zugriff.
- **Extranet**: “Schale” um ein Intranet, um beispielsweise Dienste nach außen anzubieten.
- **Internet**: Weltweites hierarchisches Netzwerk von Rechnern und Geräten.

Aufgabe 74 ★⋆

REK-AK **Datentransfer**

Ein Student möchte am Abend einen DVD-Film (Single-Layer-DVD – 4,7 GB, DVD-5, nicht kopiergeschützt) bei sich in der WG auf dem PC schauen. Allerdings stellt er fest, dass er diesen bei einem Freund liegengelassen hat. Dieser Freund wohnt im Wohnheim, das eine 36-Megabit-Internet-Anbindung hat und das er in 15 min erreichen kann (eine Richtung). Seine eigene WG hat einen DSL-Anschluss mit 16-Megabit-Internet-Anbindung.

a) Ist es schneller, wenn der Freund ihm die DVD über das Internet schickt oder wenn er die DVD bei ihm im Wohnheim abholt?

Weitere Annahmen:

- kein Overhead im Netzwerk oder auf der DVD,
- kein anderer Netzverkehr,
- keine Aufenthaltszeit bei dem Freund und
- Übertragungsrate zu/von den Laufwerken stellt keinen Flaschenhals dar.

Lösung:

Es ist schneller, den Datenträger abzuholen, denn:

$$\frac{4{,}7\ \text{GB}}{16\frac{\text{Megabit}}{\text{s}}} = \frac{4{,}7\ \text{GB}}{\frac{16\ \text{MB}}{8\ \text{s}}} = 2350\ \text{s} \approx 39{,}17\,\text{min} > 2 \cdot 15\,\text{min}$$

b) Der Student will eigentlich heute nicht mehr aus dem Haus gehen. Wird er sich ärgern, dass er letzte Woche den Vertreter seines Internetproviders weggeschickt hat, der ihm eine Umstellung auf einen VDSL2-Anschluss mit 25-Megabit-Anbindung verkaufen wollte?

Lösung:

Datentransfer über das Internet wäre in diesem Fall schneller (zum Missfallen des Studenten), denn:

$$\frac{4{,}7\ \text{GB}}{25\frac{\text{Megabit}}{\text{s}}} = \frac{4{,}7\ \text{GB}}{\frac{25\ \text{MB}}{8\ \text{s}}} = 1504\ \text{s} \approx 25{,}07\,\text{min} < 2 \cdot 15\,\text{min}$$

Erinnerung: 1 GB = 1.000 MB = 1.000.000 kB = 1.000.000.000 Byte

Aufgabe 75 ⋆

VER-AD **Rechnerarchitektur**

Geben Sie an, ob folgende Aussagen wahr oder falsch sind.

Lösung:

	Wahr	Falsch
Die Zeit, die benötigt wird, um auf einen Datensatz an einer bestimmten Adresse einer Festplatte zuzugreifen, ist für dieselbe Festplatte immer die gleiche.	☐	☒
Hyperthreading gaukelt dem Betriebssystem mehr Prozessorkerne vor als tatsächlich vorhanden sind.	☒	☐
Beim Hyperthreading werden Programme mit wenigen Threads in mehr Threads zerlegt, bevor sie ausgeführt werden.	☐	☒
Busse haben immer genau zwei angeschlossene Geräte, die Daten untereinander austauschen.	☐	☒
Seitenfehler bringen Betriebssysteme gewöhnlich zum Absturz.	☐	☒
Beim “preemptive Scheduling” kann ein Prozess so lange über den Prozessor verfügen, wie er möchte.	☐	☒

7 Programmierung

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=364

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 76 ★

PRO-AA **Programmierebenen**

Vervollständigen Sie den unten angegebenen Lückentext mithilfe der folgenden Begriffe:

Abstraktionsgrade	abstrakte Maschine	Maschinenbefehlszyklus
Interpretierung	Mikroprogrammierung	Software
Übersetzung	objektorientierte Programmiersprache	

Hinweis: Es müssen nicht alle genannten Begriffe verwendet werden.

Lösung:

Um die Arbeit des Programmierers etwas angenehmer zu gestalten, wurden im Laufe der Zeit verschiedene Programmierebenen eingeführt, die durch unterschiedliche Abstraktionsgrade von der Maschinensprache an gegebene Problemstellungen angepasst sind. Somit gibt es heute eine Hierarchie von Sprachebenen, deren Programme durch Übersetzung ineinander überführbar sind. Durch Interpretierung können diese Programme direkt ausgeführt werden. Dabei stellt sich jede Ebene nach außen als abstrakte Maschine dar. Die unterste Programmierebene ist die Mikroprogrammierung. Diese ist in der Regel jedoch nicht frei zugänglich für den Programmierer.

Aufgabe 77 ★⋆

PRO-AB **Programmiersprachen**

Sind folgende Aussagen wahr oder falsch?

Lösung:

	Wahr	Falsch
Die Programmiersprachenebenen kann man von höhersprachlich zu hardwarenah folgendermaßen sortieren: Objektorientiere Sprachen → Assemblersprachen → Mikroprogrammierung → Maschinensprachen → Hardware.	☐	☒

Erklärung: Eine richtige Sortierung wäre: problemorientierte Sprachen (imperative, objektorientierte, funktionale, prädikative) → Assemblersprachen → Maschinensprachen → Mikroprogrammierung.

Problemorientierte Programmiersprachen sind weitgehend unabhängig von der Hardware der Rechenmaschinen.	☒	☐
Befehle in Assemblersprachen werden nicht binär, sondern symbolisch notiert.	☒	☐
Durch Übersetzung sind Programme aus höheren Sprachebenen in niedrigere überführbar. Durch Interpretierung sind Programme aller Sprachenebenen direkt ausführbar.	☒	☐
Sowohl die Ebene der Maschinensprachen, als auch die der Mikroprogrammierung sind dem Anwender nicht zugänglich.	☐	☒

Erklärung: Die Ebene der Maschinensprachen ist dem Anwender zugänglich, die Ebene der Mikroprogrammierung üblicherweise nicht.

Eine Maschinensprache stellt Befehle in binärer Kodierung dar.	☒	☐
Problemorientierte Programmiersprachen können folgendermaßen kategorisiert werden: iterative Programmiersprachen, funktionale und additive Programmiersprachen, prädikative und operative Programmiersprachen.	☐	☒

Erklärung: Man kategorisiert in imperative, objektorientierte, funktionale und prädikative Programmiersprachen.

Programme in prädikativen Programmiersprachen, bestehen aus Mengen von Fakten und Regeln. Ein Beispiel aus dieser Klasse von Programmiersprachen ist PROLOG.	☒	☐

Aufgabe 78 ⋆

PRO-AC **Interpreter, Kompilierungsarten**

Nennen Sie die vier Phasen eines Kompilierers und beschreiben Sie diese kurz.

Lösung:

1) Lexikalische Analyse (**scanning**):
   - Zerlegung des Programms in Tokens
2) Syntaktische Analyse (**parsing**):
   - Überprüfung der Korrektheit des Programms
   - Erzeugung des syntaktischen Ableitungsbaums entsprechend der zur Programmiersprache gehörenden kontextfreien Grammatik
3) Semantische Analyse und Codegenerierung:
   - Überprüfung semantischer Regeln (Typenverträglichkeit usw.)
   - Erzeugung des Assembler-/Maschinenprogramms aus dem Syntaxbaum
4) Codeoptimierung:
   - Laufzeitoptimierung
   - Speicherplatzoptimierung

Aufgabe 79 ⋆

PRO-AD **Adressierungsarten**

Adressierungsarten beschreiben den von einem Programm vorgegebenen Weg, wie der Prozessor Operanden für eine Rechenoperation aus dem Speicher lädt und wie der Speicherort für das Ergebnis angegeben wird. Gegeben sei der folgende Speicher:

Adresse	Datum
0	0
1	19
2	13
3	4
4	8
5	4
6	2
7	7
8	3
9	5
10	17
11	1
12	14
13	10
14	6
15	12

Ebenfalls gegeben sei folgendes Indexregister: $\boxed{4}$

a) Welcher Wert wird bei folgenden Aufrufen geladen?

- Lade Register indirekt mit 11.
- Lade Register direkt mit 9.
- Lade Register indiziert mit 1.
- Lade Register unmittelbar mit 10.

Lösung:

- Lade Register indirekt mit 11 $\Rightarrow$ 19.
- Lade Register direkt mit 9 $\Rightarrow$ 5.
- Lade Register indiziert mit 1 $\Rightarrow$ 4.
- Lade Register unmittelbar mit 10 $\Rightarrow$ 10.

b) Im ersten Teil dieser Aufgabe wurden vier der sechs typischen Adressierungsarten abgefragt. Welche beiden weiteren Adressierungsarten kennen Sie noch?

Lösung:

Relative Adressierung und implizite Adressierung fehlen noch.

Aufgabe 80 ★⋆

PRO-AE **Maschinenbefehlszyklus**

Bei der Ausführung eines Befehls im Rechner führt das Steuerwerk den sogenannten Maschinenbefehlszyklus aus.

a) Aus welchen drei Phasen besteht der Maschinenbefehlszyklus und welche Aufgaben hat das Steuerwerk in den einzelnen Phasen?

Lösung:

- **Holen von Befehlen (FETCH)**: Der Befehlszähler PC (program counter) des Steuerwerks enthält die Adresse des aktuellen Befehls im Speicher. Dieser aktuelle Befehl wird anschließend adressiert und der Inhalt der adressierten Speicherzelle wird ins Instruktionsregister IR geladen.
- **Entschlüsseln von Befehlen (DECODE)**: Der Inhalt von IR wird analysiert und die Befehlsart wird erkannt. Der Befehl wird in den **Operationsteil** (“Was soll gemacht werden?” – Beispielsweise Addition, Subtraktion, . . .) und den **Operandenteil** (“Mit welchen Werten soll die Operation durchgeführt werden?”) zerlegt.
- **Initiieren der Befehlsausführung (EXECUTE)**: Die für die Befehlsausführung benötigten Funktionseinheiten werden mit den notwendigen Steuersignalen versorgt:
  - Adressierung der beteiligten Operanden und Laden aus dem Speicher,
  - Ausführung der Operation,
  - Speichern des Ergebnisses und
  - Hochzählen des Befehlszählers (PC).

b) Nun soll folgender umgangssprachlich formulierter Befehl konkret umgesetzt werden:

```
Subtrahiere 50 vom Inhalt der Speicherzelle 114
und speichere das Ergebnis in Speicherzelle 114 ab.
```

Geben Sie an, was das Steuerwerk in den einzelnen Schritten des Maschinenbefehlszyklus tun muss, um die Aufgabe zu erfüllen. Gehen Sie von folgendem Anfangszustand aus:

- Inhalt von PC: 347
- Inhalt von Speicherzelle 347: der zu verarbeitende Befehl
- Inhalt von Speicherzelle 114: 895

Lösung:

- **FETCH**: Der Inhalt von Speicherzelle 347 (vom Befehlszähler PC adressiert) wird ins Instruktionsregister IR geladen.
- **DECODE**: Das Steuerwerk erkennt, dass die Konstante 50 vom Inhalt der Speicherzelle 114 subtrahiert werden soll.
- **EXECUTE**: Die Operanden werden geholt und der Befehl ausgeführt:
  - der Inhalt von Speicherzelle 114 wird geholt,
  - die Subtraktion wird durchgeführt (895 − 50),
  - das Ergebnis (845) wird in Speicherzelle 114 geschrieben und
  - der Befehlszähler PC wird auf den nächsten Wert 348 hochgezählt.

Aufgabe 81 ★⋆

ASS-AF **Assembler**

Schreiben Sie ein Programm in Assemblersprache, basierend auf 1-Adress-Befehlen, welches die Funktion

$$f(n) = \sum_{i=1}^{n} i^2$$

für $n \in \mathbb{N}_0$ berechnet. Nehmen Sie dabei an, dass anfangs n in R1 und 0 in allen anderen Registern Rm ($m > 1$) gespeichert ist. Der Funktionswert $f(n)$ soll nach der Berechnung in R2 gespeichert sein. Alle anderen Register Rm ($m > 2$) können als Zwischenspeicher genutzt werden.

Lösung:

Programm in Assemblersprache für f (mit # wird unmittelbare Adressierung gekennzeichnet):

```
        LOAD        R1
        JUMPZERO    END     // Spezialfall n = 0 berücksichtigen.
        LOAD        #1
        STORE       R3      // 1 initial in R3 speichern.

LOOP    JUMPZERO    END     // Schleifenabbruch.
        LOAD        R3
        MULTIPLY    R3      // i^2 im Akkumulator zwischenspeichern.
        ADD         R2      // Bisher berechnete Summe dazu addieren...
        STORE       R2      // ...und in R2 speichern.

        LOAD        R3
        ADD         #1
        STORE       R3      // i um eins erhöhen.

        LOAD        R1
        SUB         #1
        STORE       R1      // Schleifenzähler um eins verringern.

        JUMP        LOOP
END
```

Aufgabe 82 ★⋆

ASS-AB **Assembler**

Schreiben Sie ein Assembler-Programm, basierend auf 1-Adress-Befehlen, welches die Funktion

$$f : \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0 : f(c, d, r, s) = r + \left\lfloor \frac{s \cdot (c - d)}{r + s} \right\rfloor$$

berechnet (für $r + s = 0$ kann sich das Programm beliebig verhalten). Nehmen Sie dabei an, dass anfangs c in R1, d in R2, r in R3 und s in R4 gespeichert sind. $f(c, d, r, s)$ soll abschließend in R5 gespeichert werden. Alle übrigen Register Rn ($n > 5$), können als Zwischenspeicher verwendet werden.

Lösung:

Programm in Assemblersprache für f:

```
LOAD    R1
SUB     R2
STORE   R6
LOAD    R3
ADD     R4
STORE   R7
LOAD    R4
MUL     R6
DIV     R7
ADD     R3
STORE   R5
```

Aufgabe 83 ⋆⋆

ASS-AD **Assembler**

a) Gegeben sei folgendes Assemblerprogramm basierend auf 1-Adress-Befehlen (für unmittelbare Adressierung wird das Symbol # benutzt, überall sonst wird direkt adressiert):

```
        LOAD    #1
        STORE   R3
        LOAD    #1
        STORE   R4
        LOAD    R1
        JUMPZERO END
        SUBTRACT #1
LOOP    JUMPZERO END
        SUBTRACT #1
        STORE   R1
        LOAD    R3
        STORE   R2
        LOAD    R4
        STORE   R3
        ADD     R2
        STORE   R4
        LOAD    R1
        JUMP    LOOP
END
```

Welche Funktion $f : \mathbb{N} \to \mathbb{N}$ berechnet das Assemblerprogramm, wenn zu Beginn der Wert n in R1 und nach der Ausführung der Wert $f(n)$ in R4 steht?

Lösung:

Die Funktion $f(n)$ berechnet die n-te Fibonacci-Zahl:

$$f(0) = 1, f(1) = 1; f(n) = f(n-2) + f(n-1), n > 1$$

b) Schreiben Sie ein Assemblerprogramm, das eine Zahl x als Eingabe in R1 erhält und als Ergebnis die dezimale Quersumme $Q(x)$ dieser Zahl in R3 ausgibt. Beispielsweise soll gelten: $Q(x) = Q(4112) = 8$.

Hinweis: Verwenden Sie die Befehle *DIVIDE* bzw. *MODULO*, die die ganzzahlige Division bzw. den ganzzahligen Divisionsrest berechnen.

Lösung:

Programm in Assemblersprache:

```
      LOAD     #0
      STORE    R3
      LOAD     R1
LOOP  JUMPZERO END
      DIV      #10
      STORE    R2
      LOAD     R1
      MODULO   #10
      ADD      R3
      STORE    R3
      LOAD     R2
      STORE    R1
      JUMP     LOOP
END
```

Aufgabe 84 ★⋆

ASS-AA **Assembler**

Gegeben sei das folgende Assembler-Programm auf Basis von 1-Adress-Befehlen:

```
        LOAD      # 1
        STORE     R3
        LOAD      R2
LOOP    JUMPZERO  END
        LOAD      R3
        MULTIPLY  R1
        STORE     R3
        LOAD      R2
        SUBTRACT  #1
        STORE     R2
        JUMP      LOOP
END
```

Welche Funktion $f : \mathbb{N}\times\mathbb{N} \to \mathbb{N} : (a, b) \mapsto f(a, b)$ berechnet das Programm, wenn a zu Beginn in Register R1 und b in R2 steht, und das Ergebnis in R3 abgefragt wird? (Für unmittelbare Adressierung wird das Symbol # benutzt, überall sonst wird direkt adressiert.)

Lösung:

Es wird die Funktion $f(a, b) = a^b$ berechnet.

Aufgabe 85 ★⋆

ASS-AC **Assembler**

Gegeben sei folgendes auf 1-Adress-Befehlen basierendes Assemblerprogramm:

```
        LOAD      R1
        JUMPZERO  End
        STORE     R2
        STORE     R3
LOOP    LOAD      R2
        SUBTRACT  # 1
        JUMPZERO  END
        STORE     R2
        ADD       R3
        STORE     R3
        JUMP      LOOP
END
```

Für unmittelbare Adressierung wird das Symbol # benutzt, überall sonst wird direkt adressiert.

a) Was gibt das Programm in R3 aus, wenn zu Beginn R1= 4 bzw. R1= 5 gilt?

Lösung:

$R3 = 10$ bzw. $R3 = 15$.

b) Durch welche Funktion $f : \mathbb{N} \to \mathbb{N}$ lässt sich der Endwert von R3 in Abhängigkeit vom Ausgangswert von R1 ausgeben?

Lösung:

$f(n) = \sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}, n \in \mathbb{N}_0$, wenn anfangs R1= n gilt.

Aufgabe 86 ★⋆

ASS-AE **Assembler**

Die Syntax einer auf 3-Adress-Befehlen basierenden Assemblersprache sei wie folgt aufgebaut:

$$\text{OpCode } Q_1, (Q_2,)\, Z$$

Dabei sind Q_1 und Q_2 Quellregister, die eingelesen werden und Z ein Zielregister, in das das Ergebnis der Operation gespeichert wird. Wenn eine Adresse fehlt, werden die verbleibenden zwei als Q_1 und Z behandelt.
Geben sei das folgende Assemblerprogramm (für unmittelbare Adressierung wird das Symbol # benutzt, überall sonst wird direkt adressiert):

```
      STORE       R1      R2
LOOP  SUBTRACT    R1  #1  R1
      ADD         R2  R1  R2
      JUMPNOTZERO R1      LOOP
```

a) Der Wert $f(n)$ welcher Funktion $f : \mathbb{N} \to \mathbb{N}$ in Abhängigkeit der Eingabe $n \in \mathbb{N}$, die zu Beginn in $R1$ stehe, wird am Ende in $R2$ gespeichert?

Lösung:

$f(n) = \sum_{i=1}^{n} i = 1/2 \cdot n \cdot (n+1)$

b) Schreiben Sie nun ein Assemblerprogramm, das die gleiche Funktion f berechnet, aber ohne Jump-Befehle auskommt, also konstante, statt linearer Ausführungszeit benötigt.

Lösung:

Wegen

$$f(n) = \sum_{i=1}^{n} i = 1/2 \cdot n \cdot (n + 1)$$

und weil $n \cdot (n + 1)$ immer eine gerade Zahl ist:

```
Store     R1      R2
Add       R1  #1  R1
Multiply  R2  R1  R2
Divide    R2  #2  R2
```

8 Betriebssysteme

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=365

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 87 ★⋆

BET-AD **Betriebssysteme**

a) Geben Sie eine kurze Definition eines Betriebssystems an.

Lösung:

Definition Betriebssystem: Gesamtheit aller Systemprogramme einer Rechenanlage. Sie übernehmen anfallende Verwaltungs-, Steuerungs- und Dienstleistungsaufgaben für den Prozessor, den Speicher, die E/A-Geräte sowie die Programme und Daten.

b) Nennen und erklären Sie mindestens drei Aspekte der Sicherheit, die ein Betriebssystem gewährleisten muss.

Lösung:

Aspekte der Sicherheit, die ein Betriebssystem gewährleisten muss:

- **Vertraulichkeit**: Das Betriebssystem muss Dienste bereitstellen, die es ermöglichen, den Zugriff auf Informationen auf spezifizierte Kommunikationspartner bzw. Benutzer einzuschränken (beispielsweise durch Verschlüsselung).
- **Zugriffsschutz**: Das Betriebssystem muss gewährleisten, dass kein unberechtigter Zugriff auf Daten oder auf Dienste möglich ist.
- **Nachvollziehbarkeit**: Das Betriebssystem muss gewährleisten, dass Zugriffe auf Daten oder Dienste nachvollziehbar sind (beispielsweise durch Logging).
- **Verfügbarkeit**: Das Betriebssystem muss gewährleisten, dass auch beim Ausfall von Komponenten die vom Betriebssystem zu erbringenden Dienste für den Benutzer bereitstehen.
- **Zuverlässigkeit**: Das Betriebssystem muss garantieren, dass die übernommenen Aufgaben korrekt entsprechend den spezifizierten Anforderungen erfüllt werden.

c) Nennen Sie fünf typische Dienste von Betriebssystemen.

Lösung:

Typische Dienste von Betriebssystemen:

- **Prozessorverwaltung** (Scheduling),
- **Hauptspeicherverwaltung**,
- **Programmausführung** (Laden der Befehle und Daten in den Hauptspeicher),
- **Zugriff auf E/A-Geräte** (Umsetzung allgemeiner Lese-und Schreiboperationen in gerätespezifische Steuersignale),
- **Netzzugriff**,
- **Zugriff auf Dateien**,
- **Systemzugriff**,
- **Buchführung (Systemlog)**,
- usw.

d) Das Betriebssystem behandelt auszuführende Aufträge als Prozesse. Was versteht man unter Prozessen und wozu werden sie benötigt?

Lösung:

Ein Prozess ist der zeitliche Ablauf einer Folge von Aktionen eines Rechners, die zu einer identifizierbaren funktionellen Einheit zusammengefasst sind. Er wird durch das auszuführende Programm, die zu verwendenden Daten und den Ausführungskontext charakterisiert. Ein Vorteil und die Motivation der Aufteilung eines Auftrags in mehrere Prozesse ist, dass dadurch Aufträge zeitlich verzahnt ausgeführt werden können.

e) Welche Zustände kann ein Prozess annehmen? Erklären Sie mindestens drei dieser Zustände näher.

Lösung:

Zustände, die ein Prozess annehmen kann:

- **Initiiert**: Der Prozess ist dem Betriebssystem bekannt und in den Zwischenspeicher abgelegt, aber noch nicht zur Bearbeitung zugelassen.
- **Bereit**: Der Prozess besitzt mit Ausnahme des Prozessors alle zu seiner Bearbeitung notwendigen Betriebsmittel.
- **Aktiv**: Der Prozess hat zusätzlich Prozessorzugriff und wird vom Prozessor ausgeführt (höchstens ein Prozess kann gleichzeitig in diesem Zustand sein, es sei denn, es gibt mehrere Prozessoren).
- **Blockiert**: Der Prozess wartet auf den Eintritt eines bestimmten Ereignisses (beispielsweise Abschluss einer E/A-Operation oder Zuteilung eines Betriebsmittels).
- **Terminiert**: Der Prozess hat seine Berechnungen beendet und die von ihm belegten Betriebsmittel freigegeben.

f) Ist ein Zustandsübergang von **bereit** nach **blockiert** möglich?

Lösung:

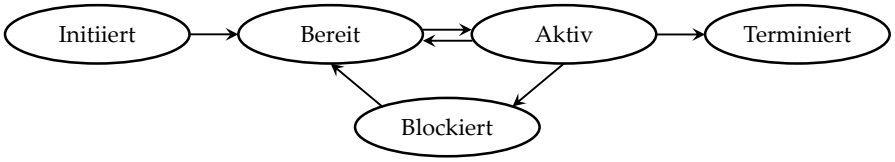
Nein, ein direkter Übergang von **bereit** nach **blockiert** ist nicht möglich. Der Prozess muss zunächst den Zustand **aktiv** erreichen, bevor er blockiert werden kann.

g) Geben Sie alle sechs möglichen Übergänge zwischen Prozesszuständen und jeweils die dafür zuständige Betriebssystemkomponente an.

Lösung:

Mögliche Zustandsübergänge mit zuständiger Betriebssystemkomponente:

- **Initiiert → bereit:** Auftragssteuerung (job scheduler),
- **Bereit → aktiv:** Prozessorverwaltung (CPU scheduler),
- **Aktiv → bereit:** Prozessorverwaltung (CPU scheduler),
- **Aktiv → blockiert:** Betriebsmittelverwaltung,
- **Aktiv → terminiert:** Betriebsmittelverwaltung,
- **Blockiert → bereit:** Betriebsmittelverwaltung.



Aufgabe 88 ⋆

BET-AB **Betriebsarten**

Welche Betriebsarten von Rechenanlagen gibt es? Nennen Sie zu jeder Betriebsart jeweils mindestens zwei typische zugehörige Merkmale.

Lösung:

- Stapelbetrieb (batch mode):
  - Motivation: möglichst gute Nutzung der Komponenten eines Rechners.
  - Merkmale:
    - Aufträge müssen vor Übergabe an die Rechenanlage inklusive aller Eingabedaten vollständig definiert sein.
    - Ein Auftrag wird immer zusammenhängend übergeben (beispielsweise in Form einer Systemdatei).

    - Kein modifizierendes Eingreifen während der Bearbeitung eines Auftrags möglich ("off-line"-Ausführung).
    - Keine Zeitanforderungen hinsichtlich der Beendigung der Aufträge möglich (aber eventuell hinsichtlich des Beginns).
    - Maximierung des Durchsatzes ist Hauptzweck (Durchsatz = Anzahl der pro Zeiteinheit vollständig bearbeiteten Aufträge).
  - Typische Anwendungen: Heute nur noch rechen- und/oder datenintensive Programme ohne Interaktion mit dem Benutzer, beispielsweise statistische Auswertungen, technisch/wissenschaftliche Berechnungen usw.

- Multiprogrammbetrieb (multiprogramming mode):
  - Merkmale:
    - Mehrere Prozesse können gleichzeitig im Hauptspeicher resident sein.
    - Bearbeitung der Prozesse zeitlich verzahnt (quasi-parallel oder nebenläufig).
    - Hauptaufgaben sind die Zuteilung von Betriebsmitteln an Prozesse, die Aktivierung und Blockierung von Prozessen und die gegenseitige Isolation der Prozesse (zur Vermeidung unerwünschter gegenseitiger Beeinflussung beim Zugriff auf gemeinsam benutzte Betriebsmittel).
    - Vorteil: bessere Ausnutzung der Betriebsmittel einer Rechenanlage.

- Dialogbetrieb (time sharing mode):
  - Merkmale:
    - Ein Auftrag wird erst im Verlauf einer Dialogsitzung interaktiv definiert (Reaktion auf Ergebnisse abgeschlossener Teilaufträge möglich).
    - Man unterscheidet zwei Arten von Dialogbetrieb:
      - Teilhaberbetrieb: alle Benutzer verwenden dasselbe Programm.
      - Teilnehmerbetrieb: die Benutzer können der Rechenanlage unabhängig voneinander verschiedene Aufträge übergeben.
    - Hauptaufgabe des Timesharing-Betriebssystems: ähnlich wie beim Multiprogrammbetrieb, jedoch schnellere Bedienung der Benutzer, also Minimierung der Antwortzeit (= Wartezeit des (Teil-) Auftrags auf Beginn/Fortsetzung seiner Bearbeitung plus Bearbeitungszeit durch Prozessor).
  - Typische Anwendungen:
    - Großrechner mit vielen Terminals.
    - Teilhaberbetrieb: Platzbuchungssystem eines Reisebüros, Programme für Schalterterminals einer Bank.
    - Teilnehmerbetrieb: interaktive Programmentwicklung.

- Echtzeitbetrieb (real-time mode):

  - Merkmale: im Prinzip wie Dialogbetrieb, jedoch zusätzlich:
    * Einhaltung von Zeitrestriktionen, es kann also für die Bearbeitung eines Auftrags eine Höchstdauer vorgegeben werden.
    * Zuteilung von Betriebsmitteln an Aufträge aufgrund ihrer Priorität (Maß für Dringlichkeit).
    * Je nach Anwendung geeignete E/A-Geräte (Messfühler, Steuerungsventile, “Sensoren”, “Aktoren” usw.) erforderlich (s. u.).
    * Unterbrechungsmöglichkeit bei der Auftragsbearbeitung (weniger dringlicher Auftrag muss zugunsten eines dringenderen unterbrochen werden können).
    * Hauptaufgabe ist eine garantierte größtmögliche und zeitgerechte Verfügbarkeit der Betriebsmittel einer Rechenanlage (also bei Anforderung soll das entsprechende Betriebsmittel sofort frei sein).
  - Typische Anwendungen:
    * Mess- und Regelungsaufgaben bei technischen Prozessen,
    * Kfz-Elektronik,
    * Telekommunikation.

- Client/Server-Betrieb (verteilte Aufgaben):
  - Merkmale:
    * Im Server werden zentrale Systemfunktionen bereitgestellt, beispielsweise:
      · Benutzerverwaltung,
      · Dateiverwaltung,
      · Datenbank-Management,
      · Mail-Server,
      · WWW-Server und
      · Druckersteuerung.
    * Jeder Rechner (“Client”) kann Funktionen des Servers durch “Remote Procedure Calls” (RPC) aufrufen.
    * Während der Ausführung eines RPCs kann der Client andere Aufgaben bearbeiten.
    * Sind Rechner sowohl Clients als auch Server, so heißen sie auch Peers.
  - Typische Anwendung: Nutzung von Diensten über das Internet.

Aufgabe 89 ★⋆

BET-AC **Betriebssysteme**

Bei der Synchronisation von Prozessen in einem Betriebssystem können Probleme auftreten. Einerseits können falsche Ergebnisse erzielt werden, andererseits birgt die Synchronisation auch die Gefahr von Deadlocks in sich.

a) Nennen Sie zwei prinzipielle Möglichkeiten, auf welche Weise Prozesse synchronisiert werden können. Bei welcher dieser Möglichkeiten stellen Deadlocks ein Problem dar?

Lösung:

Arten der Synchronisation:

- **Prozesskooperation**: Prozesse, die voneinander abhängig sind, kennen einander und koordinieren sich gegenseitig.
- **Wechselseitiger Ausschluss**: Der Zugriff eines Prozesses, welcher von anderen abhängig ist, verhindert den Zugriff anderer Prozesse auf das Betriebsmittel.

  Der wechselseitige Ausschluss kann zu einem Deadlock führen. Es handelt sich dabei um eine gegenseitige Blockade voneinander abhängiger Prozesse. So könnte beispielsweise Prozess 1 auf Betriebsmittel 2 warten, während er Betriebsmittel 1 belegt, und Prozess 2 auf Betriebsmittel 1 warten, während er Betriebsmittel 2 belegt. Das ist ein zentrales Problem in Betriebssystemen.

b) Erläutern Sie die Ursache falscher Ergebnisse und beziehen Sie sich in diesem Zusammenhang vor allem auf den kritischen Bereich.

Lösung:

Ursache falscher Ergebnisse ist, dass kritische Bereiche von Prozessen nicht beachtet werden. Der kritische Bereich ist eine Folge von Operationen, in denen ein Prozess nicht unterbrochen werden darf. Kritische Bereiche entstehen, wenn mehrere Prozesse um das gleiche Betriebsmittel konkurrieren. Solche Prozesse nennt man auch voneinander abhängig.

Aufgabe 90 ⋆

BET-AA **Betriebssysteme**

Gegeben sei folgendes Forschungsszenario, in dem eine Rechenanlage genutzt wird:

> Ein Team erforscht die Faltung von Proteinen, die sehr rechenintensiv ist. Dafür erstellt jeder Forscher eigene Programme und gibt diese an die Rechenanlage. Jeder Forscher erhält so viel Rechenzeit, wie sein Programm benötigt, da nur die vollständig errechneten Daten verwendbar sind. Mit diesen arbeitet der Forscher eine Zeitlang weiter, bis er ein neues Programm der Rechenanlage übergibt.

Welche Betriebsart würden Sie für die Rechenanlage in diesem Szenario vorschlagen? Begründen Sie kurz.

Lösung:

Stapelbetrieb (batch mode), da einzelne Programmabläufe keine Interaktion benötigen, keine zeitlichen Beschränkungen gefordert sind und maximaler Durchsatz erwünscht ist.

Aufgabe 91 ⋆

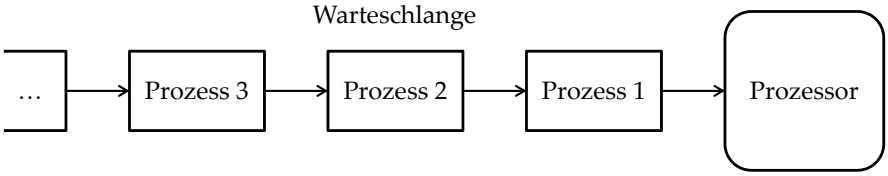
BET-AE **Verfahren zur Zuteilung von Rechenzeit**

Nennen und erläutern Sie kurz drei Typen von Verfahren zur Zuteilung von Rechenzeit.

Lösung:

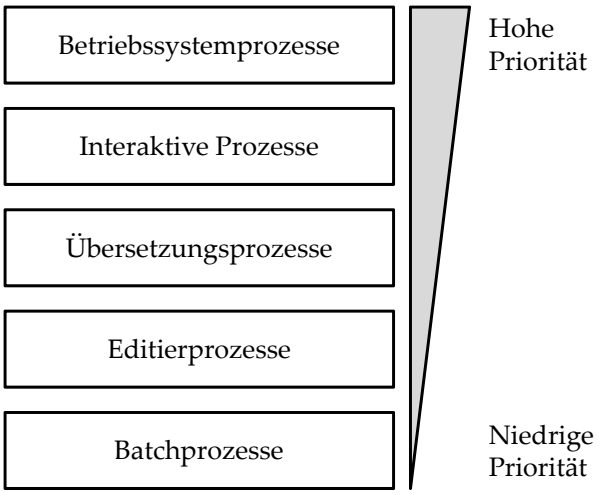
Verfahren zur Zuteilung von Rechenzeit:

- **Einfache Zuteilungsverfahren**, beispielsweise “first-come, first-serve” (FCFS). Das Prinzip ist, möglichst viel Prozessorleistung für Benutzerprozesse und möglichst wenig für Zuteilungsprozesse bereitzustellen. Bei FCFS erhält daher einfach der erste Prozess, der anfragt, als erster Rechenzeit:



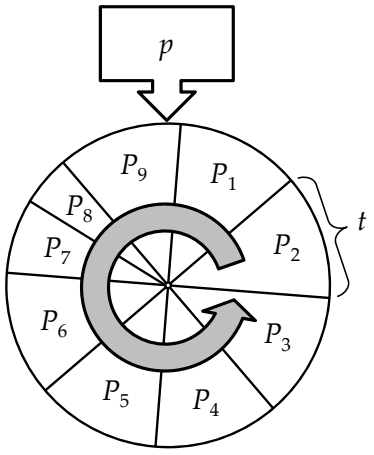
- **Prioritätsgesteuerte Zuteilungsverfahren**.
  - Beispiel eines einfachen prioritätsgesteuerten Zuteilungsverfahrens: weise jedem bereiten Prozess eine Priorität zu und aktiviere immer den Prozess mit der höchsten Priorität.

– Beispiel eines komplizierteren prioritätsgesteuerten Zuteilungsverfahrens: teile die Menge der bereiten Prozesse in Klassen unterschiedlicher Priorität ein, beispielsweise:



– Weitere Möglichkeiten der Prioritätszuweisung sind "shortest job first" (erzielt sehr gute mittlere Wartezeiten), eine Kombination von Wartezeit und erwarteter Laufzeit usw.

- **Zeitscheibenverfahren** (auch "**Round-Robin**"). Die Menge der bereiten Prozesse wird zirkulär durchlaufen. Pro Durchlauf erhält jeder Prozess genau einmal höchstens die fest vorgegebene Prozessorzeit t und wird danach in der Ausführung bis zum nächsten Durchlauf unterbrochen. Benötigt ein Prozess nur die Zeit $t' < t$, wird vorzeitig mit dem nächsten Prozess fortgefahren (vgl. P_7, P_8 in der Abbildung). Das Verfahren wird auch **preemptive Scheduling** genannt, da ein Prozess vor seiner Fertigstellung unterbrochen werden kann, falls $t' > t$. Im Gegensatz dazu, wird ein Verfahren, bei dem ein Prozess nie vor seiner Fertigstellung unterbrochen werden kann, **nonpreemptive Scheduling** genannt. (Vgl. auch Lösungen zu Aufgaben BET-AF und BET-AG.)



Aufgabe 92 ★⋆

BET-AG **Verfahren zur Zuteilung von Rechenzeit**

Gegeben sei eine Menge $\prod = \{P_1, P_2, P_3, P_4, P_5\}$ von bereiten Prozessen und deren benötigte CPU-Zeiten $t(P_i)$ für $i \in \{1, \ldots, 5\}$, die in folgender Tabelle dargestellt sind:

Prozessnummer i	Benötigte CPU-Zeit $t(P_i)$
1	33 ms
2	17 ms
3	48 ms
4	9 ms
5	66 ms

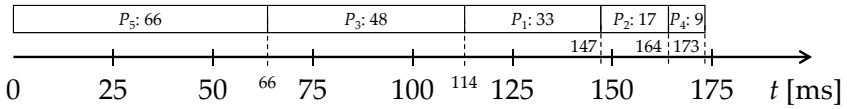
a) Für zwei Prozesse P_i, P_j mit $1 \leq i \neq j \leq 5$ seien die Prioritäten $p(P_i), p(P_j)$ für die Dringlichkeit der Abarbeitung der Prozesse relativ zueinander wie folgt gegeben ("longest job first"):

$$p(P_i) \geq p(P_j) \Leftrightarrow t(P_i) \geq t(P_j)$$

Teilen Sie die Rechenzeit mit einem prioritätsgesteuerten Zuteilungsverfahren den Prozessen zu, indem Sie einen Zeitstrahl mit der Zuteilung beschriften (gehen Sie bei dieser Aufgabe davon aus, dass die Kontextwechsel vernachlässigbar kurze Zeit dauern).

Lösung:

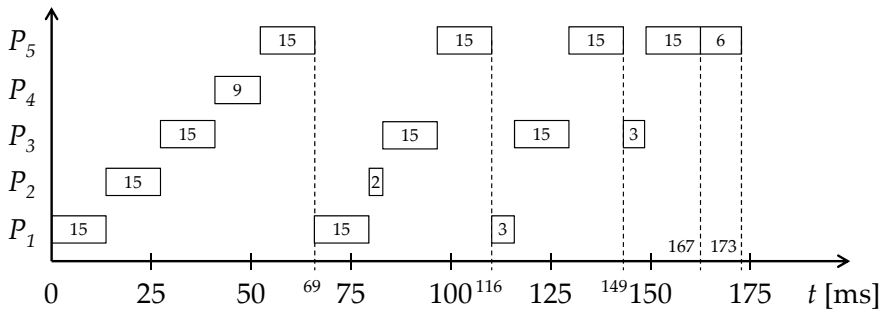
Prioritätsgesteuertes Zuteilungsverfahren:



b) Verwenden Sie nun das Zeitscheibenverfahren mit einer maximalen Zeitspanne pro Prozess von 15 ms, um die Rechenzeit auf die Prozesse zuzuteilen und tragen Sie das Ergebnis auf einem Zeitstrahl ein (gehen Sie auch hier davon aus, dass die Kontextwechsel vernachlässigbar kurze Zeit dauern). Die Prozesse werden in der Reihenfolge $P_1 \to P_2 \to P_3 \to P_4 \to P_5 \to P_1 \ldots$ durchlaufen.

Lösung:

Zeitscheibenverfahren:



c) Erläutern Sie die Unterschiede zwischen preemptive Scheduling und nonpreemptive Scheduling. Welches dieser beiden Verfahren charakterisiert das prioritätsgesteuerte Zuteilungsverfahren und welches das Zeitscheibenverfahren (Round-Robin)?

Lösung:

Beim **preemptive Scheduling** kann ein Prozess vor der Fertigstellung unterbrochen und später wieder fortgesetzt werden. Beim **nonpreemptive Scheduling** kann ein Prozess dagegen nicht vorzeitig abgebrochen bzw. unterbrochen werden. (Vgl. auch Lösungen zu Aufgaben BET-AE und BET-AF.) Prioritätsgesteuerte Zuteilungsverfahren gehören zum nonpreemptive Scheduling, Zeitscheibenverfahren zum preemptive Scheduling.

Aufgabe 93 ★⋆

BET-AF **Zeitscheibenverfahren**

Das Zeitscheibenverfahren, bzw. Round-Robin-Verfahren ist eine Möglichkeit der Zuteilung von Rechenzeit.

a) Erklären Sie kurz die Arbeitsweise dieses Verfahrens.

Lösung:

Allen Prozessen, die sich im Wartebereich befinden (also den Prozesszustand **bereit** einnehmen), wird eine maximale feste Zeitspanne ("time-slice") zugeordnet. Nach Ablauf der maximalen Zeitspanne (spätestens) wird dem Prozess der Prozessor entzogen und dem nächsten zugeteilt. Ein solches Verfahren wird auch als **preemptive Scheduling** bezeichnet, weil ein Prozess vor seiner Fertigstellung unterbrochen werden kann. (Vgl. auch Lösungen zu Aufgaben BET-AE und BET-AG.)

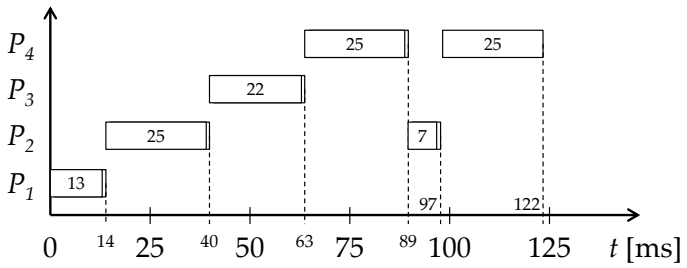
b) Sei Z eine Zeitscheibe aus dem Round-Robin-Verfahren mit **Gesamtdauer** 100 ms pro Umdrehung. Weiterhin gebe es vier Prozesse P_1, P_2, P_3 und P_4, die folgende Zeiten $t(P_1), \ldots, t(P_4)$ zur Fertigstellung benötigen:

- $t(P_1) = 13\,\text{ms}$,
- $t(P_2) = 32\,\text{ms}$,
- $t(P_3) = 22\,\text{ms}$ und
- $t(P_4) = 50\,\text{ms}$.

Die Zeit für einen Kontextwechsel betrage 1 ms. Die Prozesse werden in der Reihenfolge $P_1 \to P_2 \to P_3 \to P_4 \to P_1 \ldots$ durchlaufen. Stellen Sie den Ablauf der Abarbeitung der Prozesse grafisch dar und berechnen Sie den durch die Kontextwechsel erzeugten Overhead, also die Zeit, die durch die Wechsel nicht für die Prozesse, sondern für das Zuteilungsverfahren genutzt wird.

Lösung:

Ablauf der Abarbeitung der Prozesse:



Auf die vier Prozesse wird die Gesamtzeit von 100 ms verteilt, indem die “time-slice” auf 25 ms gesetzt wird. Der erzeugte Overhead, also der Zeitverlust durch die Prozesswechsel, beträgt 5 ms, da es bei sechs Prozessabschnitten insgesamt fünf Kontextwechsel gibt.

9 Dateiorganisation

→ Fragen und Antworten zu diesem Kapitel

http://info2.aifb.kit.edu/qa/index.php?qa=366

Falls Sie Fragen zu einer **bestimmten Aufgabe** haben, klicken Sie bitte auf die Aufgaben-ID unter dem entsprechenden Aufgabentitel (von der Form **PRO-BE**).

«Kapitel-Zusammenfassungen sind in dieser Voransicht nicht verfügbar»

Aufgabe 94 ★

DAT-AF **Dateiorganisation**

Inwiefern unterscheidet sich die Sekundärorganisation (SO) von der Primärorganisation (PO) und was sind die Aufgaben dieser Dateiorganisationsformen?

Lösung:

- Die PO legt fest, wie interne Datensätze auf dem Speichermedium abgelegt werden und wie sie wiedergefunden werden können. Die PO bestimmt also die Anordnung von Sätzen bei der physischen Speicherung (Erstabspeicherung). Man unterscheidet unter anderem sequentielle und gestreute POs.
  - Eine sequentielle PO ordnet zusammenhängende Folgen von Sätzen entweder unsortiert oder sortiert direkt hintereinander auf dem Speichermedium an. Im unsortierten Fall (auch "seriell") wird der Primärschlüssel nicht beachtet, sondern beim Einfügen einfach der Eingabereihenfolge gefolgt. Im sortierten Fall wird die Reihenfolge durch einen Primärschlüssel definiert.
  - Eine gestreute PO sucht nach geeigneten freien Speicherplätzen ohne auf eine physisch zusammenhängende Speicherung der Sätze zu achten. Hierzu kann beispielsweise eine Hash-Organisation, die den Speicherplatz mittels einer Funktion berechnet, oder eine Index-sequentielle Organisation, bei der der Speicherplatz mittels einer Tabelle bestimmt wird, verwendet werden.
- Die SO beeinflusst die physische Anordnung der Sätze nicht. Sie bietet eine "andere Sicht" auf eine bereits bestehende solche Anordnung. Dies geschieht beispielsweise durch eine geeignete Datenstruktur (Zugriffspfad). Gegebenenfalls unterscheidet sie Sekundärschlüssel nach Schlüssel und Nichtschlüssel (Sekundärschlüssel müssen keine Schlüssel sein), um je nach Fall unterschiedliche Verfahren anzuwenden.

Aufgabe 95 ⋆

DAT-AD **Dateiorganisation**

a) Nach welchen Kriterien wird die Effizienz von Dateiorganisationsformen beurteilt?

Lösung:

- Aufwand für das Lesen und Schreiben eines Satzes (insbesondere sequentieller vs. wahlfreier Zugriff);
- Aufwand für das Einfügen und Entfernen eines Satzes (inklusive Aufwand für Verwaltung der Dateiorganisation);
- Speicherplatzausnutzung (inkl. Speicherplatz für Verwaltungsinformationen);
- ...

b) Nennen Sie wichtige Anforderungen, die Datenverarbeitungssysteme erfüllen sollten.

Lösung:

- Die Speicherung der Daten auf externen Speichermedien, also auf großen, nichtflüchtigen Speichern mit Sicherungsmöglichkeiten (d. h. Sicherheitskopien) oder Übertragungsmöglichkeiten auf andere Datenverarbeitungsanlagen sollte möglich sein;
- Jedes Datum sollte schnell verfügbar sein, auch wenn große Datenmengen vorhanden sind.
- Der Änderungsdienst für das Hinzufügen, Ändern und Entfernen von Daten sollte schnell und zuverlässig sein;
- Die Strukturierung zusammengehörender Daten zu einer Einheit, beispielsweise Kundendaten nach einzelnen Kunden oder gesamtem Kundenbestand, sollte möglich sein;
- Wichtige Sicherheitsaspekte sollten erfüllt werden; dazu gehört die Verwaltung von Zugriffsrechten und das Führen eines Datei-Logbuchs;
- ...

Aufgabe 96 ★⋆

DAT-AB **Dateiorganisation, Hash-Organisation**

a) Nennen Sie die Haupttypen, der Primärorganisation von Daten und erläutern Sie diese kurz. Ordnen Sie in diesem Zusammenhang insbesondere die Hash-Organisation passend ein.

Lösung:

Bei den Formen der Primärorganisation (PO) von Dateien unterscheidet man die sequentielle Primärorganisation von der gestreuten PO. Bei der sequentiellen PO werden die Sätze physisch zusammenhängend (also sequentiell einer nach dem anderen) auf dem Speichermedium abgespeichert. Dies kann unsortiert (gemäß der Eingabe-Reihenfolge, ohne Beachtung des Primärschlüssels) oder sortiert (Reihenfolge durch Primärschlüssel gegeben) erfolgen. Bei der gestreuten PO ist die Folge der Sätze nicht notwendigerweise physisch zusammenhängend. Stattdessen wird nach einem passenden freien Speicherplatz im gesamten Speicherraum oder einem Teil davon gesucht. Die Hash-Organisation zählt zur Gruppe der gestreuten POs. Der Speicherort wird hierbei durch eine Funktion berechnet. Zu den gestreuten POs gehört auch die Index-sequentielle Organisation, bei der der Speicherplatz mittels einer Tabelle bestimmt wird.

b) Was wird auf Hardware-Seite benötigt, damit eine Hash-Organisation möglich ist? Welche zusätzlichen Anforderungen, werden an eine Hash-Funktionen gestellt?

Lösung:

Da bei einer Hash-Organisation die Erstabspeicherung der Sätze auf einem beliebigen freien Speicherplatz erfolgen kann (die Erstabspeicherung ist also nicht notwendigerweise sequentiell, sondern gestreut), muss, um dies effizient zu ermöglichen, ein direkter/wahlfreier Zugriff auf den Speicher gegeben sein.

An eine Hash-Funktion h selbst werden die folgenden Anforderungen gestellt:

- Die Funktion muss surjektiv sein, das bedeutet, dass sie den gesamten Adressraum des Datenspeicherraums der Datei abbildet.
- Sie soll effizient berechenbar sein.
- Alle $h^{-1}(a)$ mit $a \in AR(D)$ sollen etwa gleichmächtig sein.

c) Gegeben sei ein Speicher der Größe $M = 5$ mit den relativen Satznummern (RSN) $0, \ldots, 4$.

RSN	Satz mit Schlüssel
0	
1	
2	
3	
4	

In der folgenden Tabelle seien zudem die Primärschlüssel von Datensätzen gegeben, die in den Speicher aufgenommen werden sollen. Bei einer Kollision soll lineares Austesten erfolgen.

Primärschlüssel s	46	29	98	60	162
$h(s)$					

- Definieren Sie allgemein eine passende Hash-Funktion h unter der Berücksichtigung des gegebenen Bild- und Wertebereichs.
- Gegeben sei nun die Hash-Funktion

$$h(s) = s \bmod 5$$

  Füllen Sie die beiden Tabellen aus.

Lösung:

- Zum Beispiel $h : \mathbb{N} \to \{0, 1, 2, 3, 4\} : s \mapsto s \bmod k, k \leq M = 5$.
- Für $h(s) = s \bmod 5$ ergeben sich folgende Tabellen:

Schlüssel s	46	29	98	60	162
$h(s)$	1	4	3	0	2

RSN	Satz mit Schlüssel
0	60
1	46
2	162
3	98
4	29

In diesem Fall sind keine Kollisionen aufgetreten.

Aufgabe 97 ★⋆

DAT-AE **Hash-Organisation, Index-sequentielle Organisation**

Vergleichen Sie Effizienz der Hash-Organisation und die der Index-sequentiellen Organisation relativ zueinander in Bezug auf die folgenden Kriterien.

Lösung:

	Hash-Organisation	Index-sequentielle Organisation
Sequentielles Lesen	Sehr schlecht	I. A. gut
Sequentielles Schreiben	Sehr schlecht	I. A. gut
Wahlfreies Lesen	Im Mittel sehr gut (Zeit O(1)), im schlechtesten Fall sehr schlecht (Zeit $O(N)$ bzw. $O(M)$)	Schnell (falls keine langen Überlaufketten)
Wahlfreies Schreiben	im Mittel sehr gut (Zeit O(1)), im schlechtesten Fall sehr schlecht (Zeit $O(N)$ bzw. $O(M)$)	Schnell (falls keine langen Überlaufketten)
Satz einfügen	Sehr gut (wenn Belegung gering)	Gut (u. U. Index aktualisieren)
Satz entfernen	Sehr gut (logisches Entfernen)	Schnell (falls keine langen Überlaufketten)
Speicherplatznutzung	Weniger gut	Weniger gut

Logisches Entfernen bedeutet, dass Daten als gelöscht und überschreibbar markiert werden. N ist die Anzahl der Blöcke des verfügbaren Speicherraums, M die der relativen Satznummern (RSN).

Aufgabe 98 ★

DAT-AC **Hash-Organisation**

a) Was ist eine Hash-Funktion und wie werden Hash-Funktionen im Zusammenhang mit Dateiorganisation eingesetzt?

Lösung:

Eine Hash-Funktion ist eine effizient berechenbare, surjektive Abbildung; das bedeutet, dass der Bildbereich kleiner sein kann als der Wertebereich. Insbesondere hat das zur

Folge, dass mehrere Eingabewerte auf denselben Funktionswert abgebildet werden können bzw. dass die Umkehrabbildung $h^{-1} : B \to A$ einer Hash-Funktion $h : A \to B$ einen Wert $b \in B$ auf eine Menge von Werten $A' \subset A$ abbilden kann. In der Dateiorganisation werden Hash-Funktionen vor allem für die gestreute Organisation von Sätzen eingesetzt.

b) Sie werden gebeten, eine Hash-Funktion auszuwählen, auf deren Grundlage Sätze in einem Speicher abgelegt werden sollen. Aufgrund von sonstigen Beschränkungen bleiben Ihnen zwei Möglichkeiten:

- $h_1 : \mathbb{N} \to \mathbb{N} : h_1(x) = x \mod 17$ oder
- $h_2 : \mathbb{N} \to \mathbb{N} : h_2(x) = x \mod 10$

Welche der beiden Hash-Funktionen würden Sie auswählen? Begründen Sie kurz.

Lösung:

h_1, da Hash-Funktionen immer so gewählt werden sollten, dass die Schlüssel möglichst gleichmäßig verteilt sind, dass also für alle $b \in B$ die Menge $f^{-1}(b)$ ungefähr gleich mächtig ist. Man kann sich überlegen, dass bei Hash-Funktionen, die über die Modulo-Funktion berechnet werden, dafür Primzahlen als Divisor besonders gut geeignet sind.

Aufgabe 99 ★

DAT-AA **Hash-Organisation**

In einen Speicher der Größe $M = 8$ sollen nacheinander die folgenden Schlüssel aufgenommen werden:

$$64 - 7 - 62 - 70 - 83 - 19 - 6 - 99$$

Zur Einsortierung eines Schlüssels s wird die folgende Hash-Funktion verwendet:

$$h(s) = s \bmod M = s \bmod 8$$

Als Kollisionsbehandlungsstrategie wird lineares Austesten verwendet.

a) Wie sieht der Speicherblock nach dem Einfügen der Werte aus (RSN steht für relative Satznummer)?

Lösung:

Speicherblock:

RSN	Satz mit Schlüssel
0	64
1	70
2	6
3	83
4	19
5	99
6	62
7	7

b) Begründen Sie, warum man diese Hash-Funktion in der Praxis nicht verwenden würde und geben Sie eine bessere Hash-Funktion an.

Lösung:

Statt $M = 8$ wähle man beispielsweise $M = p$ für eine Primzahl p; die verbesserte Hash-Funktion ist dann weiterhin $h(s) = s \bmod M$ mit dem neuen M. Auf diese Weise werden die Schlüssel gleichmäßiger verteilt.

Aufgabe 100 ★⋆

DAT-AG **Hash-Organisation**

Gegeben sei für einen Primärschlüssel $s \in \mathbb{N}$ folgende Hash-Funktion $h : \mathbb{N} \to \mathbb{N}$:

$$h(s) = (s \cdot a + b) \bmod 11$$

Dabei gelte zunächst

$$a = b = 3$$

Zudem seien die folgenden Primärschlüssel von Datensätzen gegeben, die in einem Speicher der Größe 11 abgelegt werden sollen:

135 – 102 – 28 – 61 – 6 – 14 – 94

a) Berechnen Sie die Hashwerte der gegebenen Primärschlüssel.

Lösung:

Die Hashwerte lauten wie folgt

s	135	102	28	61	6	14	94
$h(s)$	1	1	10	10	10	1	10

b) Fügen Sie die Datensätze in der angegebenen Reihenfolge in den durch folgende Tabelle dargestellten Speicher ein (RSN steht für relative Satznummer), indem Sie die Primärschlüssel in die entsprechenden Positionen eintragen. Verwenden sie bei Kollisionen die Strategie des linearen Austestens.

Lösung:

Speicher bei Vermeidung von Kollisionen durch lineares Austesten:

RSN	Primärschlüssel s	Datensatz
0	61	...
1	135	...
2	102	...
3	6	...
4	14	...
5	94	...
6		...
7		...
8		...
9		...
10	28	...

c) Fügen Sie die Datensätze nun aufsteigend sortiert in den durch folgende Tabelle dargestellten Speicher ein. Verwenden sie bei Kollisionen wieder die Strategie des linearen Austestens.

Lösung:

Speicher bei Vermeidung von Kollisionen durch lineares Austesten:

RSN	Primärschlüssel s	Datensatz
0	28	...
1	14	...
2	61	...
3	94	...
4	102	...
5	135	...
6		...
7		...
8		...
9		...
10	6	...

d) Fügen Sie die Datensätze nun wieder in der gegebenen (nicht aufsteigenden) Reihenfolge in den durch folgende Tabelle dargestellten Speicher ein. Verwenden Sie in Kollisionsfällen diesmal die Strategie der separaten Verkettung.

Lösung:

Speicher nach separater Verkettung:

RSN	Primärschlüssel s	Verkettungen								
0		→		→		→		→		→
1	135	→	102	→	14	→		→		→
2		→		→		→		→		→
3		→		→		→		→		→
4		→		→		→		→		→
5		→		→		→		→		→
6		→		→		→		→		→
7		→		→		→		→		→
8		→		→		→		→		→
9		→		→		→		→		→
10	28	→	61	→	6	→	94	→		→

e) Lässt sich die Anzahl der Kollisionen bzw. die Länge der Verkettungen in d), durch andere Werte für a und b reduzieren?

Lösung:

Nein. Durch die lineare Funktion $s \cdot a + b$ können sich zwar für verschiedene Werte für a und b die Ergebnisse von $h(s)$ verändern, eine der Eigenschaften der Modulo-Funktion ist jedoch, dass die Gruppierungen sich durch eine solche Funktion nicht weiter unterteilen lassen; es werden folglich weiterhin die gleichen Schlüssel kollidieren. Für bestimmte a und b fallen sogar bisher getrennte Mengen zu einer größeren Kollisionsmenge zusammen. Beispielsweise ergibt sich für $a = 11$, $b = 0$ stets das Ergebnis $h(s) = 0$.

f) Verändern sich durch eine andere Wahl für a und b die RSN-Zeilen, auf denen es zu Kollisionen kommt?

Lösung:

Da sich die Ergebnisse von $h(s)$ durch verschiedene Werte für a und b verändern können, kann es dazu kommen, dass die Kollisionen einer auf diese Art veränderten Hash-Funktion auf anderen RSN-Zeilen auftreten.