Materialien zur Vorbereitung auf das 1. Staatsexamen Informatik (Theoretische Informatik) in Bayern

FSI Lehramt Informatik (Erlangen)

20. August 2024

Lizenz

Inhaltsverzeichnis

1 Vorwort

1.1 Aufbau und Motivation

Ziel dieses Dokuments und des zugehörigen Repositorys ist es, eine studentische Sammlung an Lösungsvorschlägen zu den Aufgaben vergangener 1. Staatsexamen im Fach Informatik in Bayern zu erstellen. Wir wollen dadurch kooperative und nachhaltige Vorbereitung fördern. Da wir leider kein Urheberrecht an den Aufgabenstellungen haben, können wir diese jedoch nicht veröffentlichen. Wir bitten um Verständnis.

Als Ausgleich dafür soll diese Sammlung auch einen Überblick über die Themenbereiche der Informatik geben, die in den Staatsexamen abgefragt werden. Dabei soll idealerweise eine Sammlung an Definitionen, Algorithmen, Sätzen und Ähnlichem entstehen, die potenziell mit eigens erstellten Beispielen ergänzt werden.

1.2 Mitmachen

von Max Mustermann

`E-Mail: max@mustermann.de`

Wir freuen uns selbstverständlich über jede Art von Beitrag. Sei es das Hinzufügen von Lösungen, das Korrigieren von Fehlern oder das Ergänzen von Inhalten in den Themenbereichen. Ausführliche Informationen zum Mitwirken lassen sich auf Github finden.

Wir möchten auch die Möglichkeit geben, eigene Lösungsvorschläge für Aufgaben mit den eigenen Autor-Informationen zu versehen. Dies sieht man beispielsweise an diesem Unterkapitel. Dies ist aber komplett optional!

2 Themenbereiche

2.1 Reguläre Sprachen

Definition 2.1.1 (Reguläre Grammatik)**.** Eine Grammatik $G = (V, \Sigma, P, S)$ heißt *regulär*, alle durch P induzierten Regeln von einer der folgenden Formen sind:

$$\begin{aligned} A &\to aB && A, B \in V, a \in \Sigma \\ A &\to a && a \in \Sigma \\ A &\to \epsilon \end{aligned}$$

Definition 2.1.2 (Reguläre Sprache)**.** Eine Sprache L heißt *regulär*, wenn es eine reguläre Grammatik G gibt, sodass $L = L(G)$.

Theorem 2.1.3 (Pumping-Lemma)**.** *Sei L eine formale Sprache. Wenn L regulär ist, dann gibt es eine Zahl $n \geq 1$, sodass für jedes Wort $w \in L$ mit $|w| \geq n$ drei Teilwörter $x, y, z \in \Sigma^*$ existieren, sodass*

$$w = xyz \qquad \text{(PL-Partition)}$$
$$|xy| \leq n \qquad (2.1)$$
$$y \neq \epsilon \qquad (2.2)$$
$$\forall p \in \mathbb{N}.\, xy^p z \in L \qquad \text{(PL-Pumping)}$$

Beispiel 2.1.4. Die Sprache aus (2.7) ist nicht regulär. Dies lässt sich zum Beispiel durch einen Widerspruchsbeweis mittels Pumping-Lemma zeigen.

Sei $n \geq 1$ die Pumping-Zahl. Es gilt

$$w := a^n b c^n \in L$$
$$|w| = 2n + 1 > n$$

Es gibt also $x, y, z \in \Sigma^*$, sodass (PL-Partition), (2.1), (2.2) und (PL-Pumping) gelten.

Wegen (2.1) folgt, dass ein $k \in \mathbb{N}$ existiert, sodass

$$y = a^k \qquad (2.3)$$

Wegen (2.2) und (2.3) folgt direkt

$$k \geq 1 \qquad (2.4)$$

Wegen (PL-Pumping) müsste nun auch gelten, dass

$$w' := xy^0z \in L \tag{2.5}$$

Dies ist aber nicht der Fall, wie folgende Rechnung zeigt:

$$\begin{aligned} w' &= xy^0z \\ &= xz \\ &= a^{n-k}bc^n \end{aligned}$$

Aus (2.4) folgt $n - k \neq n$, und damit ist $w' \notin L$, was im Widerspruch zur Aussage des Pumping-Lemmas steht. Folglich kann L nicht regulär sein.

2.2 Kontextfreie Sprachen

Definition 2.2.1 (Kontextfreie Grammatik)**.** Eine Grammatik $G = (V, \Sigma, P, S)$ heißt *kontextfrei*, wenn alle durch P induzierten Regeln von folgender Form sind:

$$A \to \alpha \quad A \in V, \alpha \in (V \cup \Sigma)^* \tag{2.6}$$

Definition 2.2.2 (Kontextfreie Sprache)**.** Eine Sprache L heißt *kontextfrei*, wenn es eine kontextfreie Grammatik G gibt, sodass $L = L(G)$.

Beispiel 2.2.3. Die Sprache

$$L := \{a^kbc^l \in \{a, b, c\}^* \mid k, l \in \mathbb{N}, k \geq l\} \tag{2.7}$$

ist kontextfrei. Dies lässt sich durch eine kontextfreie Grammatik zeigen, die L erzeugt:

$$\begin{aligned} G_L &:= (V, \Sigma, P, S) \\ V &:= \{S\} \\ \Sigma &:= \{a, bkc\} \\ P&: \\ &S \to b \mid aS \mid aSc \end{aligned}$$

Ein Beweis, dass tatsächlich $L = L(G_L)$ gilt, ist noch zu führen.

Beispiel 2.2.4. Sei $G := (\{S\}, \{x, y\}, P, S)$ mit folgenden Produktionen P:

$$S \to yxx \mid xSyx \mid SS$$

Man kann nun zeigen, dass für alle Wörter $w \in L(G)$ gilt, dass $|w|_x = 2|w|_y$. $w \in L(G)$ bedeutet, dass

$$S \Longrightarrow^* w \tag{2.8}$$

Um (2.8) klassifizieren zu können, bietet sich folgende (noch zu beweisende) Hilfsaussage an:

$$\forall w \in \{S, x, y\}^*. S \Longrightarrow^* w \to |w|_x = 2|w|_y \tag{2.9}$$

Mithilfe von (2.9) lässt sich direkt für alle $w \in L(G)$ folgern, dass $|w|_x = 2|w|_y$.

Beweis von (2.9). Ein induktiver Beweis über das Prädikat (2.8) bietet sich an.
Sei also zunächst $S \Longrightarrow^0 S$ gegeben. Es gilt:

$$|S|_x = 0 = |S|_y \tag{2.10}$$

Sei nun $S \Longrightarrow^* w' \Longrightarrow w$ gegeben, wobei bereits folgende Induktionsvoraussetzung gilt:

$$|w'|_x = 2|w'|_y \tag{IV}$$

Aus $w' \Longrightarrow w$ folgt, dass w' noch aus mindestens einem Nichtterminal besteht, also noch ein S enthält. Sei nun v das von diesem S abgeleitete Teilwort. Wegen (2.10) ergeben sich nun folgende Zusammenhänge:

$$|w|_x = |w'|_x + |v|_x \tag{2.11}$$
$$|w|_y = |w'|_y + |v|_y \tag{2.12}$$

Nach den Produktionsregeln kann v nur von einer der folgenden Formen sein:

$$v = yxx$$
$$v = xSyx$$
$$v = SS$$

In allen 3 Fällen lässt sich leicht nachrechnen, dass

$$|v|_x = 2|v|_y \tag{2.13}$$

Damit ergibt sich folgende Rechnung:

$$\begin{aligned}
|w|_x &= |w'|_x + |v|_x && \text{(Wegen (2.11))} \\
&= 2|w'|_y + |v|_x && \text{(Wegen (IV))} \\
&= 2|w'|_y + 2|v|_y && \text{(Wegen (2.13))} \\
&= 2(|w'|_y + 2|v|_y) && \text{(Arithmetik)} \\
&= 2|w|_y && \text{(Wegen (2.12))}
\end{aligned}$$

□

2.3 Entscheidbarkeit

Definition 2.3.1 (Entscheidbarkeit)**.** Sei $L \subseteq \Sigma^*$ eine formale Sprache.

1. Eine TM M *entscheidet* eine L, wenn für alle $w \in \Sigma^*$ gilt, dass
   - M auf w hält,
   - $w \in L \Longleftrightarrow w \in L(M)$
2. L heißt *entscheidbar*, wenn es eine TM M gibt, die L entscheidet.

Definition 2.3.2 (Semi-Entscheidbarkeit)**.** Sei $L \subseteq \Sigma^*$ eine formale Sprache.
L heißt *semi-entscheidbar*, wenn es eine TM M gibt, sodass für alle $w \in \Sigma^*$ gilt, dass $w \in L \Longleftrightarrow w \in L(M)$.

Bemerkung 2.3.3. Semi-Entscheidbarkeit unterscheidet sich also von Entscheidbarkeit dadurch, dass die TM M nur zwingend auf w hält, wenn $w \in L$.

Theorem 2.3.4 (Gleichmächtigkeit von DTM und NTM)**.** *Zu jeder NTM M gibt es eine DTM M' mit $L(M) = L(M')$.*

Beispiel 2.3.5. Sei $L := \{c(M) \mid M \text{ berechnet } f\colon \mathbb{N} \to \mathbb{N}, \text{sodass } \exists n \in \mathbb{N}.\, f(n) = n!\}$.

Semi-Entscheidbarkeit: Dieses Problem ist semi-entscheidbar. Dies lässt sich durch Angabe einer 2-Band-NTM beweisen, die L berechnet.

Sei also M eine TM. Gesucht: Eine 2-Band-NTM D, die M genau dann akzeptiert, wenn M eine Funktion $f\colon \mathbb{N} \to \mathbb{N}$ berechnet, sodass ein $n \in \mathbb{N}$ existiert, sodass $f(n) = n!$.

D soll sich wie folgt verhalten:

1. Zunächst schreibt D nicht-deterministisch eine natürliche Zahl auf beide Bänder. Auf beiden Bändern soll die gleiche Zahl stehen.
2. Anschließend wird M auf dieser Eingabe auf dem ersten Band gestartet.
3. Wenn M hält: Schreibe auch $n!$ auf das Band rechts daneben. Dies ist sicher möglich, da die Fakultätsfunktion LOOP-berechenbar ist.
4. Nun lässt sich durch Vergleich überprüfen, ob die beiden Zahlen gleich sind.

   Falls beide Zahlen gleich sind, halte akzeptierend.

   Falls beide Zahlen ungleich sind, halte nicht-akzeptierend.

Untscheidbarkeit: L ist jedoch unentscheidbar. Dies lässt sich wie folgt zeigen:

$$\begin{aligned} L &= \{c(M) \mid M \text{ berechnet } f\colon \mathbb{N} \to \mathbb{N}, \text{sodass } \exists n \in \mathbb{N}.\, f(n) = n!\} \\ &= \{c(M) \mid M \text{ berechnet } f \in \{f\colon \mathbb{N} \to \mathbb{N} \mid \exists n \in \mathbb{N}.\, f(n) = n!\}\} \end{aligned}$$

Zu zeigen ist nun, dass für $S := \{f\colon \mathbb{N} \to \mathbb{N} \mid \exists n \in \mathbb{N}.\, f(n) = n!\}$ gilt:

$$S \neq \emptyset \qquad \text{(Not Empty)}$$
$$S \neq R \qquad \text{(Not Full)}$$

Dabei ist R die Menge aller Turing-berechenbaren Funktionen. (Not Empty) lässt sich mit dem Zeugen $f_1(n) := n!$ belegen, (Not Full) lässt sich mit dem Zeugen $f_2(n) := 0$ belegen.

Damit folgt mittels des Satzes von Rice, dass L unentscheidbar ist.

Beispiel 2.3.6. Sei $L := \{c(M) \mid M \text{ TM, die auf } 1111001 \text{ hält}\}$.

Unentscheidbarkeit: L ist zunächst unentscheidbar, wie sich durch Reduktion auf das allgemeine Halteproblem zeigen lässt. Das Halteproblem ist wie folgt definiert:

$$L_{halt} := \{c(M)w \,|\, M \text{ TM, die auf } w \text{ hält}\} \tag{2.14}$$

Sei $f\colon \Sigma^* \to \Sigma^*$ definiert durch

$$f(x) := \begin{cases} c(M') & \text{falls } x = c(M)w \begin{cases} \text{für eine TM } M = (Z, \Sigma, \Gamma, \delta, z_0, F) \\ \text{und ein Wort } w = a_1 \dots a_n \end{cases} \\ 0 & \text{sonst} \end{cases} \tag{2.15}$$

Dabei ist $M' := (Z \cup \{q_i \,|\, 0 \leq i \leq n\}, \Sigma, \Gamma, \beta, q_0, F)$, wobei β wie folgt definiert ist:

$$\begin{aligned} \beta(q_0, s) &:= (q_0, \#, R) \text{ für } s \in \Gamma \setminus \{\#\} \\ \beta(q_i, \#) &:= (q_{i+1}, a_{n-i}, L) \text{ für } 0 \leq i \leq n-1 \\ \beta(q_n, \#) &:= (z_0, \#, R) \\ \beta(z, s) &:= \delta(z, s) \text{ für } s \in \Gamma,\ z \in Z \end{aligned}$$

M' arbeitet also wie folgt:

1. Zuerst löscht M' ihre Eingabe.
2. Danach schreibt M' das Wort w auf das Band und setzt den Curser auf den Anfang des Wortes.
3. Anschließend startet M' die gegebene Turingmaschine M.

.

Zu zeigen ist nun, dass f in der Tat eine Reduktion darstellt:

- f ist total, da die Funktion offensichtlich auf jeder Eingabe definiert ist.
- f ist korrekt in dem Sinne, dass für alle Wörter $w \in \Sigma^*$ gilt, dass $w \in L_{halt}$ genau dann gilt, wenn $f(w) \in L$. Hierzu ist nach Konstruktion von M' zu bemerken, dass M' genau dann auf *irgendeine* Eingabe hält, wenn M auf w hält.

  Der Rest zeigt sich ebenso einfach:

$$\begin{aligned} x \in L_{halt} &\iff \begin{cases} x = c(M)w \\ M \text{ hält auf } w \end{cases} \\ &\iff M' \text{ hält auf jede Eingabe} \\ &\iff M' \text{ hält auf } 1111001 \\ &\iff f(w) \in L \end{aligned}$$

  Die Rückrichtung des vorletzten Schritts begründet sich ebenso durch Konstruktion von M': Wenn M' auf irgendeine Eingabe hält, dann bereits auf jede Eingabe.

- f ist berechenbar:

1. Zunächst wird ein Syntaxcheck auf x durchgeführt. Dies ist immer möglich.
2. Falls $x = c(M)w$, schreibe neue Übergänge von M' auf das Band (und halte). Dazu müssen M und w ausgewertet werden.
   Falls $x \neq c(M)w$, schreibe 0 und halte.

Damit ist f eine gesuchte Reduktion. Da L_{halt} bereits unentscheidbar ist, muss nun auch L unentscheidbar sein.

Semi-Entscheidbarkeit: L ist jedoch semi-entscheidbar. Dies lässt sich durch die Konstruktion einer deterministischer Turingmaschine belegen, die sich wie folgt verhält:

1. Schreibe neben die Gödelisierung der Eingabe M das Wort 101.
2. Starte die universelle Turingmaschine M_u auf diese neue Eingabe. Falls M_u hält: Halte und akzeptiere.

Da sich M_u auf Eingabe $c(M)101$ verhält wie M auf Eingabe 101, akzeptiert diese Turingmaschine genau dann, wenn M auf 101 hält.

Semi-Entscheidbarkeit des Komplements: Da L unentscheidbar und semi-entscheidbar ist, kann L^c nicht semi-entscheidbar und damit auch nicht entscheidbar sein.

Wären sowohl L als auch L^c semi-entscheidbar, so ließe sich eine 2-Band-TM konstruieren, wobei auf dem ersten Band der Semi-Entscheider für L und auf dem zweiten Band der Semi-Entscheider für L^c simuliert wird. Diese 2-Band-TM hält genau dann, wenn eines der beiden Bänder hält und akzeptiert. Da Band 1 akzeptiert, wenn die Eingabe Teil von L ist, und Band 2 akzeptiert, wenn die Eingabe nicht Teil von L ist, wird die 2-Band-TM immer halten und wäre damit ein Entscheider für L. Folglich kann L^c in dieser Aufgabe nicht semi-entscheidbar sein.

3 Lösungsversuche

3.1 F12T1

Aufgabe 1

von Max Ole Elliger

E-Mail: ole.elliger@fau.de

Definiere zunächst

$A_1 \coloneqq \{(i,t) \in \mathbb{N} \times \mathbb{N} \mid M_i \text{ hält auf Eingabe } i \text{ nicht innerhalb von } t \text{ Schritten}\}$
$A_2 \coloneqq \{(i,x) \in \mathbb{N} \times \mathbb{N} \mid M_i \text{ hält nicht auf die Eingabe } x\}$
$A_3 \coloneqq \{i \in \mathbb{N} \mid M_i \text{ berechnet } f\colon \mathbb{N} \to \mathbb{N} \text{ und } \exists x \in \mathbb{N}.\, f(x) = x\}$

A_1 ist entscheidbar. Folgende TM M entscheidet A_1:

1. Gegeben $(i,t) \in \mathbb{N} \times \mathbb{N}$.
2. Simuliere M_i auf Eingabe i.
3. Wenn M_i mindestens $t+1$ Schritte rechnet, halte und akzeptiere. Ansonsten halte und akzeptiere nicht.

A_2 ist co-r.e.. A_2 ist das Komplement des Halteproblems. Da das Halteproblem semi-entscheidbar (also r.e.) ist, kann A_2 nicht auch r.e. sein, ist also co-r.e.

A_3 ist r.e.. Folgende NTM N semi-entscheidet A_3:

1. Gegeben $i \in \mathbb{N}$.
2. Wähle nicht-deterministisch ein $x \in \mathbb{N}$.
3. Simuliere M_i mit Eingabe x.
4. Falls das berechnete Ergebnis wieder x ist: Halte und akzeptiere.

A_3 ist nicht entscheidbar. Es gilt:

$$A_3 = \{i \in \mathbb{N} \mid M_i \text{ berechnet } f \in \{f \mid \exists x \in \mathbb{N}.\, f(x) = x\}\}$$

Definiere nun

$$S \coloneqq \{f \mid \exists x \in \mathbb{N}.\, f(x) = x\}$$

Es gilt:

$$(x \mapsto x) \in S$$
$$(x \mapsto x + 1) \notin S$$

Mithilfe des Satzes von Rice folgt nun, dass die A_3 nicht entscheidbar ist.

Aufgabe 2

von Max Ole Elliger

E-Mail: ole.elliger@fau.de

Schreibe zunächst die Überführungsfunktion von M tabellarisch auf:

δ	a	b	final?
q_0	q_0, q_2, q_3	q_3	ja
q_1			ja
q_2	q_1		ja
q_3	q_0	q_4	nein
q_4	q_0, q_2, q_3		nein

Baue nun einen DFA M' mit $L(M) = L(M')$. Dazu ergibt sich folgende Überführungsfunktion δ':

δ'	a	b	final?
$\{q_0\}$	$\{q_0, q_2, q_3\}$	$\{q_3\}$	ja
$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_3, q_4\}$	ja
$\{q_3\}$	$\{q_0\}$	$\{q_4\}$	nein
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_3, q_4\}$	ja
$\{q_3, q_4\}$	$\{q_0, q_2, q_3\}$	$\{q_4\}$	nein
$\{q_4\}$	$\{q_0, q_2, q_3\}$	$\emptyset$	nein
$\emptyset$	$\emptyset$	$\emptyset$	nein

Die restlichen definierenden Elemente von M' ergeben sich ebenfalls aus der tabellarischen Darstellung. Insbesondere ist $\{q_0\}$ der Startzustand von M'.

3.2 F21T1

Aufgabe 1

von Max Ole Elliger

E-Mail: ole.elliger@fau.de

Zuordnung der NFA's

NFA	Sprache
N_1	L_6
N_2	L_8
N_3	L_2
N_4	L_12
N_5	L_8
N_6	L_11

Beweis der Nicht-Regularität von L_3 Angenommen, L_3 wäre regulär, dann gibt es nach Pumping-Lemma eine Zahl $n \geq 1$, sodass für jedes Wort $w \in L_3$ mit $|w| \geq n$ drei Teilwörter $x, y, z \in \Sigma^*$ existieren, sodass (PL-Partition), (2.1), (2.2) und (PL-Pumping) gelten.

Sei also n die Pumping-Zahl. Betrachte das Wort $w := a^{2^n} \in L_3$. Es gilt $|w| = 2^n \geq n$. Es gibt also $x, y, z \in \Sigma^*$, sodass (PL-Partition), (2.1), (2.2) und (PL-Pumping) gelten.

Wegen (2.1) folgt, dass ein $k \in \mathbb{N}$ existiert, sodass $y = a^k$. Wegen (2.2) folgt direkt $k \geq 1$. Wegen (PL-Pumping) müsste nun auch gelten, dass $w' := xy^2z \in L_3$. Dies ist aber nicht der Fall, wie folgende Rechnung zeigt:

$$w' = xy^2z = a^{2^n+k} \notin L_3$$

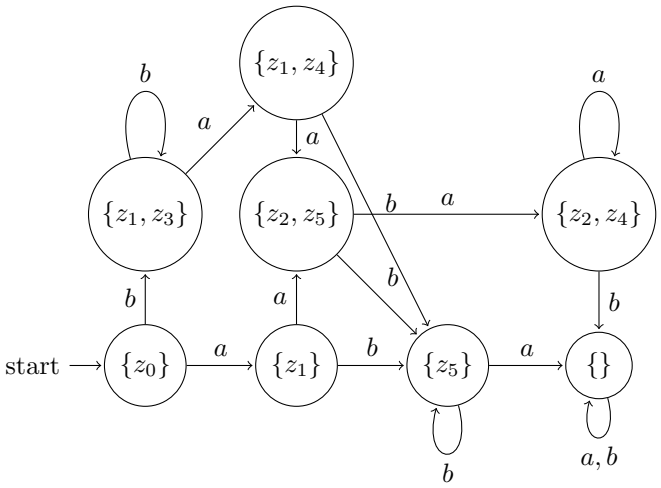
Die Folgerung im letzten Schritt ergibt sich aus der Tatsache, dass $2^n + k \neq 2^{n'}$ für alle $n' \in \mathbb{N}$ ist, denn dafür müsste $k \geq 2^n$ gelten, was wegen $k \leq n$ nicht möglich ist. Also ist $w' \notin L_3$, was im Widerspruch zur Annahme steht, dass L_3 regulär ist.

Potenzmengenkonstruktion Das Vorgehen ist wie folgt: Man startet zunächst beim neuen Startzustand $\{z_0\}$. Die Übergänge ergeben sich durch folgende Formel:

$$\forall zs \subseteq Z, a \in \Sigma.\, \delta'(zs, a) := \{z \in Z \mid \exists z' \in zs.\, \delta(z', a) = z\}$$

Die Menge der Endzustände lässt sich wie folgt berechnen:

$$F' := \{zs \in Z \mid zs \cap F \neq \emptyset\}$$



Aufgabe 3

von Max Ole Elliger

E-Mail: ole.elliger@fau.de

Die Folge $1, 1, 1, 1$ ist keine Lösung von K: Das Wort x ist länger als die Wörter y oder z, und y und z unterscheiden sich im ersten Buchstaben.

Die Folge $2, 2, 2, 2$ ist eine Lösung von K, da dann $y = z$ gilt.

PCP lässt sich auf DPCP wie folgt reduzieren:

$$f(K) := \begin{cases} (x_1, y_1, 1), \ldots, (x_k, y_k, k) & \text{falls } K = (x_1, y_1), \ldots, (x_k, y_k) \\ \epsilon & \text{sonst} \end{cases}$$

Dabei sei o.B.d.A. angenommen, dass $\Sigma \cap \mathbb{N} = \emptyset$.

f ist total Da f für jede Eingabe definiert ist, ist f total.

f ist berechenbar Ein Syntaxcheck zu Beginn ist möglich. Das Ergänzen eines Tupels ist in $O(1)$ möglich, also kann der neue Ausdruck in $O(k)$ niedergeschrieben werden.

f ist korrekt

$$\begin{aligned} K \in PCP &\iff K = \ldots \wedge \exists m.\, x = y \\ &\iff f(K) = \ldots \wedge \exists m.\, x = y \\ &\iff f(K) = \ldots \wedge \exists m.\, x = y \vee x = z \vee y = z \\ &\iff f(K) \in DPCP \end{aligned}$$

Folglich ist DPCP nicht entscheidbar.

Aufgabe 4

von Max Ole Elliger

E-Mail: ole.elliger@fau.de

Umwandlung von Klauseln mit 0 Literalen Diese Klauseln evaluieren zu 0. Folglich werden die folgenden Klauseln generiert:

$$\{\{x, y, z\} \mid x \in \{a, \neg a\}, y \in \{b, \neg b\}, z \in \{c, \neg c\}\}$$

Diese Klauseln können nicht alle gemeinsam erfüllt werden.

1 Literal Sei dieses eine Literal x. Generiere folgende Klauseln:

$$\{\{x, y, z\} \mid y \in \{b, \neg b\}, z \in \{c, \neg c\}\}$$

2 Literale Seien die beiden Literale x und y. Generiere folgende Klauseln:

$$\{\{x, y, z\} \mid z \in \{c, \neg c\}\}$$

3 Literale Kopiere die Klausel in die gerade zu generierende Genau-3-SAT-Formel.

Die Generierung der neuen Klauseln ist pro Klausel in $O(1)$ schaffbar, damit hängt die Komplexität der Umwandlung linear von der Anzahl der Klauseln ab, ist also in polynomieller Zeit berechenbar. Sie ist auch offensichtlich total und korrekt. Die Korrektheit sieht man wie folgt am Beispiel einer 0-Literal-generierten Klausel: Jede dieser Klauseln stellt eine mögliche Belegung für drei (frische) Variablen dar. Eine erfüllende Belegung kann also nicht alle Klauseln gleichzeitig erfüllen. Die Begründung für die anderen Fälle erfolgt analog.

Da $3-CNF-SAT$ zu $GENAU-3-CNF-SAT$ reduziert, ist letzteres Problem mindestens so schwer wie $3-CNF-SAT$. Da $3-CNF-SAT$ NP-vollständig, also insbesondere NP-hart ist, folgt, dass damit auch $GENAU-3-CNF-SAT$ NP-hart ist. Es fehlt also nur noch der Beweis, dass $GENAU-3-CNF-SAT$ nicht-deterministisch in polynomieller Zeit entscheidbar ist. Ein entsprechender Algorithmus verhält sich wie folgt:

1. Wähle nicht-deterministisch eine erfüllende Belegung κ.
2. Überprüfe für jede der k Klauseln, ob eines der drei Literale von κ erfüllt wird. ($O(k)$)
3. Falls das für alle Klauseln gilt, akzeptiere, ansonsten akzeptiere nicht.

Damit ist $GENAU-3-CNF-SAT$ in NP, also NP-vollständig.

- Bezüglich i): Dann wäre $GENAU-3-CNF-SAT$ in P, woraus direkt $P=NP$ folgt.
- Bezüglich ii): Dann wäre $GENAU-3-CNF-SAT$ nicht in P, woraus direkt $P \neq NP$ folgt.
- Bezüglich iii): Keine Aussage möglich.

Index